

# BIOS and Kernel Developer's Guide for AMD Athlon™ 64 and AMD Opteron™ Processors


| | |
|---|---|
| Publication # **26094** | Revision: **3.04** |
| Issue Date: **July 2003** | |

**© 2002, 2003 Advanced Micro Devices, Inc.** All rights reserved.

The contents of this document are provided in connection with Advanced Micro Devices, Inc. (“AMD”) products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD’s Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD’s products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD’s product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

**Trademarks**

AMD, the AMD Arrow logo, AMD Athlon, AMD Opteron, and combinations thereof, 3DNow!, and AMD PowerNow! are trademarks of Advanced Micro Devices, Inc.

HyperTransport is a licensed trademark of the HyperTransport Technology Consortium.

MMX is a trademark and Pentium is a registered trademark of Intel Corporation.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

# Contents

**Revision History . . . . . 15**

**1 Introduction and Overview . . . . . 17**

1.1 About This Guide . . . . . 17

1.2 Related Documents . . . . . 18

1.3 Conventions and Definitions . . . . . 18

1.3.1 Notation . . . . . 18

1.4 Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors . . . . . 19

**2 Processor Initialization and Configuration . . . . . 21**

2.1 Bootstrap Processor Initialization . . . . . 21

2.1.1 Detecting AP and Initializing Routing Table . . . . . 21

2.1.2 Initializing Noncoherent HyperTransport™ Technology Devices . . . . . 22

2.1.3 Initializing Link Width and Frequency . . . . . 22

2.1.4 Initializing the Memory Controller on All Processor Nodes . . . . . 23

2.1.5 Initializing the Address Map Table . . . . . 23

2.2 Application Processor Initialization . . . . . 23

2.3 BIOS Requirement for 64-Bit Operation . . . . . 24

2.3.1 Sizing and Testing Memory above 4 Gbytes . . . . . 24

2.3.2 Using BIOS Callbacks in 64-Bit Mode . . . . . 24

**3 Memory System Configuration . . . . . 25**

3.1 Configuration Space Accesses . . . . . 25

3.1.1 Configuration Address Register . . . . . 25

3.1.2 Configuration Data Register . . . . . 26

3.2 Memory System Configuration Registers . . . . . 27

3.3 Function 0—HyperTransport™ Technology Configuration . . . . . 27

3.3.1 Device/Vendor ID Register . . . . . 29

3.3.2 Status/Command Register . . . . . 30

3.3.3 Class Code/Revision ID Register . . . . . .30
3.3.4 Header Type Register . . . . . .31
3.3.5 Capabilities Pointer Register . . . . . .31
3.3.6 Routing Table Node *i* Registers . . . . . .32
3.3.7 Node ID Register . . . . . .34
3.3.8 Unit ID Register . . . . . .35
3.3.9 HyperTransport™ Transaction Control Register . . . . . .36
3.3.10 HyperTransport™ Initialization Control Register . . . . . .40
3.3.11 LDT*i* Capabilities Register . . . . . .42
3.3.12 LDT*i* Link Control Registers . . . . . .43
3.3.13 LDT*i* Frequency/Revision Registers . . . . . .47
3.3.14 LDT*i* Feature Capability Registers . . . . . .48
3.3.15 LDT*i* Buffer Count Registers . . . . . .49
3.3.16 LDT*i* Bus Number Registers . . . . . .51
3.3.17 LDT*i* Type Registers . . . . . .51
3.4 Function 1—Address Map . . . . . .53
3.4.1 Device/Vendor ID Register . . . . . .55
3.4.2 Class Code/Revision ID Register . . . . . .55
3.4.3 Header Type Register . . . . . .56
3.4.4 DRAM Address Map . . . . . .57
3.4.5 Memory-Mapped I/O Address Map Registers . . . . . .59
3.4.6 PCI I/O Address Map Registers . . . . . .62
3.4.7 Configuration Map Registers . . . . . .64
3.5 Function 2—DRAM Controller . . . . . .66
3.5.1 Device/Vendor ID Register . . . . . .67
3.5.2 Class Code/Revision ID Register . . . . . .68
3.5.3 Header Type Register . . . . . .69
3.5.4 DRAM CS Base Address Registers . . . . . .69
3.5.5 DRAM CS Mask Registers . . . . . .72
3.5.6 DRAM Bank Address Mapping Register . . . . . .73

3.5.7 DRAM Timing Low Register . . . . . 78
3.5.8 DRAM Timing High Register . . . . . 80
3.5.9 DRAM Configuration Registers . . . . . 82
3.5.10 DRAM Delay Line Register . . . . . 88
3.6 Function 3—Miscellaneous Control . . . . . 88
3.6.1 Device/Vendor ID Register . . . . . 90
3.6.2 Class Code/Revision ID Register . . . . . 91
3.6.3 Header Type Register . . . . . 91
3.6.4 Machine Check Architecture (MCA) Registers . . . . . 92
3.6.5 ECC and Chip Kill Error Checking and Correction . . . . . 108
3.6.6 Scrub Control Register . . . . . 110
3.6.7 DRAM Scrub Address Registers . . . . . 112
3.6.8 XBAR Flow Control Buffers . . . . . 113
3.6.9 XBAR-to-SRI Buffer Count Register . . . . . 118
3.6.10 Display Refresh Flow Control Buffers . . . . . 119
3.6.11 Power Management Control Registers . . . . . 120
3.6.12 GART Aperture Control Register . . . . . 122
3.6.13 GART Aperture Base Register . . . . . 124
3.6.14 GART Table Base Register . . . . . 124
3.6.15 GART Cache Control Register . . . . . 125
3.6.16 Clock Power/Timing Low Register . . . . . 126
3.6.17 Clock Power/Timing High Register . . . . . 128
3.6.18 HyperTransport™ FIFO Read Pointer Optimization Register . . . . . 128
3.6.19 Thermtrip Status Register . . . . . 130
3.6.20 Northbridge Capabilities Register . . . . . 131
**4 DRAM Configuration . . . . . 133**
4.1 Programming Interface . . . . . 134
4.1.1 SPD ROM-Based Configuration . . . . . 134
4.1.2 Non-SPD ROM-Based Configuration . . . . . 137
4.1.3 DRAM Initialization . . . . . 139

4.2 DRAM Configuration . . . . . 140
**5 Machine Check Architecture . . . . . 143**
5.1 Determining Machine Check Support . . . . . 143
5.2 Machine Check Errors . . . . . 143
5.2.1 Sources of Machine Check Errors . . . . . 144
5.3 Machine Check Architecture Registers . . . . . 146
5.3.1 Global Machine Check Model-Specific Registers (MSRs) . . . . . 146
5.3.2 Error Reporting Bank Machine Check MSRs . . . . . 149
5.4 Error Reporting Banks . . . . . 152
5.4.1 Data Cache (DC) . . . . . 152
5.4.2 Instruction Cache (IC) . . . . . 156
5.4.3 Bus Unit (BU) . . . . . 158
5.4.4 Load Store Unit (LS) . . . . . 161
5.4.5 Northbridge (NB) . . . . . 162
5.5 Initializing the Machine Check Mechanism . . . . . 163
5.6 Using Machine Check Features . . . . . 163
5.6.1 Handling Machine Check Exceptions . . . . . 164
**6 System Management Mode (SMM) . . . . . 167**
6.1 SMM Overview . . . . . 167
6.2 Operating Mode and Default Register Values . . . . . 167
6.3 SMM State Save Definition . . . . . 169
6.4 SMM Initial State . . . . . 171
6.5 SMM-Revision Identifier . . . . . 172
6.6 SMM Base Address . . . . . 173
6.7 Auto Halt Restart . . . . . 173
6.8 SMM I/O Trap and I/O Restart . . . . . 174
6.8.1 SMM I/O Trap . . . . . 175
6.8.2 SMM I/O Restart Byte . . . . . 175
6.9 Exceptions and Interrupts in SMM . . . . . 176
6.10 Protected SMM and ASeg/TSeg . . . . . 176

6.10.1 SMM_MASK Register . . . . . 177
6.10.2 SMM_ADDR Register . . . . . 178
6.10.3 SMM ASeg . . . . . 178
6.10.4 SMM TSeg . . . . . 179
6.10.5 Closing SMM . . . . . 180
6.10.6 Locking SMM . . . . . 180
6.11 SMM Special Cycles . . . . . 180
**7 Advanced Programmable Interrupt Controller (APIC) . . . . . 181**
7.1 Interrupt Delivery . . . . . 182
7.2 Vectored Interrupt Handling . . . . . 182
7.3 Spurious Interrupts . . . . . 183
7.3.1 Spurious Interrupts Caused by Timer Tick Interrupt . . . . . 183
7.4 Lowest-Priority Arbitration . . . . . 184
7.5 Inter-Processor Interrupts . . . . . 185
7.6 APIC Timer Operation . . . . . 185
7.7 State at Reset . . . . . 185
7.8 Register Summary . . . . . 186
7.8.1 APIC ID Register . . . . . 187
7.8.2 APIC Version Register . . . . . 187
7.8.3 Task Priority Register . . . . . 188
7.8.4 Arbitration Priority Register . . . . . 188
7.8.5 Processor Priority Register . . . . . 188
7.8.6 End of Interrupt Register . . . . . 189
7.8.7 Logical Destination Register . . . . . 189
7.8.8 Destination Format Register . . . . . 190
7.8.9 Spurious Interrupt Vector Register . . . . . 190
7.8.10 In-Service Registers . . . . . 191
7.8.11 Trigger Mode Registers . . . . . 192
7.8.12 Interrupt Request Registers . . . . . 193
7.8.13 Error Status Register . . . . . 194

7.8.14 Interrupt Command Register Low . . . . . 195
7.8.15 Interrupt Command Register High . . . . . 196
7.8.16 Timer Local Vector Table Entry . . . . . 197
7.8.17 Performance Counter Local Vector Table Entry . . . . . 197
7.8.18 Local Interrupt 0 (Legacy INTR) Local Vector Table Entry Register . . . . . 198
7.8.19 Local Interrupt 1 (Legacy NMI) Local Vector Table Entry . . . . . 199
7.8.20 Error Local Vector Table Entry . . . . . 199
7.8.21 Timer Initial Count Register . . . . . 200
7.8.22 Timer Current Count Register . . . . . 200
7.8.23 Timer Divide Configuration Register . . . . . 201
**8 HyperTransport™ Technology Configuration and Enumeration . . . . . 203**
8.1 Initial Configuration Steps . . . . . 203
8.2 One-Node Coherent HyperTransport™ Technology Initialization . . . . . 204
8.3 Two-Node Coherent HyperTransport™ Technology Initialization . . . . . 204
8.4 Generic HyperTransport™ Technology Configuration . . . . . 205
8.4.1 Algorithm . . . . . 207
**9 Power and Thermal Management . . . . . 215**
9.1 Stop Grant . . . . . 216
9.2 C-States . . . . . 218
9.2.1 C1 Halt State . . . . . 218
9.2.2 C2 and C3 . . . . . 218
9.3 Throttling . . . . . 218
9.4 Processor ACPI Thermal Zone . . . . . 219
9.5 Processor Performance States . . . . . 219
9.5.1 BIOS Requirements for P-State Transitions . . . . . 219
9.5.2 BIOS-Initiated P-State Transitions . . . . . 221
9.5.3 BIOS Support for Operating System/CPU Driver-Initiated P-State Transitions . . 221
9.5.4 Processor Driver Requirements . . . . . 222
9.5.5 P-State Transition Sequence . . . . . 222
9.6 ACPI 2.0 Processor P-State Objects . . . . . 230

9.6.1 _PCT (Performance Control) . . . . . 230
9.6.2 _PSS (Performance-Supported States) . . . . . 231
9.6.3 _PPC (Performance Present Capabilities) . . . . . 237
9.6.4 PSTATE_CNT . . . . . 239
9.6.5 CST_CNT . . . . . 239
9.7 BIOS Support for AMD PowerNow!™ Software with Legacy Operating Systems . . . 239
9.8 System Configuration for Power Management . . . . . 241
9.8.1 Chipset Configuration for Power Management . . . . . 241
9.8.2 Processor Configuration for Power Management . . . . . 241
**10 Performance Monitoring . . . . . 245**
10.1 Performance Counters . . . . . 245
10.2 Performance Event-Select Registers . . . . . 246
**11 BIOS Checklist . . . . . 255**
11.1 CPUID . . . . . 255
11.2 CPU Speed Detection . . . . . 255
11.3 HyperTransport™ Link Frequency Selection . . . . . 255
11.4 Multiprocessing Capability Detection . . . . . 256
11.5 Model-Specific Registers (MSRs) . . . . . 256
11.6 Machine Check Architecture (MCA) . . . . . 257
11.7 Memory Map . . . . . 257
11.7.1 I/O and Memory Type and Range Registers (IORRs, MTRRs) . . . . . 257
11.7.2 Memory Map Registers (MMRs) . . . . . 258
11.8 Cache Testing and Programming . . . . . 258
11.9 Memory System Configuration Registers . . . . . 258
11.10 XSDT Table . . . . . 259
11.11 Detect Target Operating Mode Callback . . . . . 259
11.12 SMM Issues . . . . . 260
**12 Processor Configuration Registers . . . . . 261**
12.1 General Model-Specific Registers . . . . . 261
12.1.1 System Software Registers . . . . . 263

12.1.2 Memory Typing Registers . . . . . .265
12.1.3 APIC Registers . . . . . .282
12.1.4 Software Debug Registers . . . . . .283
12.1.5 Performance Monitoring Registers . . . . . .284
12.2 AMD Athlon™ 64 processor and AMD Opteron™ Processor Model-Specific Registers . . . . . .284
12.2.1 Feature Registers . . . . . .286
12.2.2 Identification Registers . . . . . .291
12.2.3 Memory Typing Registers . . . . . .292
12.2.4 I/O Range Registers . . . . . .293
12.2.5 System Call Extension Registers . . . . . .294
12.2.6 Segmentation Registers . . . . . .296
12.2.7 Power Management Registers . . . . . .298
12.2.8 IO and Configuration Space Trapping to SMI . . . . . .301
**Glossary . . . . . .305**
**Index of Register Names . . . . . .311**

# List of Figures

Figure 1. Interleave Example (IntlvEn Relation to IntlvSel) ..... 59
Figure 2. Default SMM Memory Map ..... 168
Figure 3. An Eight-Node Configuration ..... 205
Figure 4. High-Level P-state Transition Flow ..... 225
Figure 5. Example P-State Transition Timing Diagram ..... 226
Figure 6. Phase 1: Core Voltage Transition Flow ..... 227
Figure 7. Phase 2: Core Frequency Transition Flow ..... 228
Figure 8. Phase 3: Core Voltage Transition Flow ..... 229

# List of Tables

Table 1. Related Documents ....18
Table 2. Function 0 Configuration Registers ....27
Table 3. Function 1 Configuration Registers ....53
Table 4. Function 2 Configuration Registers ....66
Table 5. DRAM CS Base Address and DRAM CS Mask Registers....70
Table 6. Mapping Host Address Lines to Memory Address Lines (64-Bit Interface)....74
Table 7. Mapping Host Address Lines to Memory Address Lines (128-Bit Interface)....75
Table 8. Swapped physical address lines for interleaving with 64-bit interface....77
Table 9. Swapped physical address lines for interleaving with 128-bit interface....77
Table 10. Function 3 Configuration Registers ....89
Table 11. Error Code Field Formats....99
Table 12. Transaction Type Bits (TT)....99
Table 13. Cache Level Bits (LL)....99
Table 14. Memory Transaction Type Bits (RRRR) ....99
Table 15. Participation Processor Bits (PP) ....100
Table 16. Time-Out Bit (T) ....100
Table 17. Memory or I/O Bits (II)....100
Table 18. Northbridge Error Codes....100
Table 19. Northbridge Error Status Bit Settings ....103
Table 20. ECC Syndromes ....108
Table 21. Chip Kill ECC Syndromes ....109
Table 22. Scrub Rate Control Values....111
Table 23. XBAR Input Buffers ....114
Table 24. Default XBAR Command Buffer Allocation....114
Table 25. Default Virtual Channel Command Buffer Allocation ....115
Table 26. An Example of a Non Default Virtual Channel Command Buffer Allocation ....115
Table 27. GART PTE Organization....125
Table 28. Trwt Values....138
Table 29. RdPreamble Values....139
Table 30. Sources of Machine Check Errors....145
Table 31. Valid MC0_ADDR Bits....156
Table 32. Valid MC1_ADDR Bits....158
Table 33. Valid MC2_ADDR Bits....161
Table 34. SMM Save State (Offset FE00–FFFFh) ....169
Table 35. SMM Entry State....172
Table 36. SMM ASeg-Enabled Memory Types....179
Table 37. SMM TSeg-Enabled Memory Types....179
Table 38. APIC Register Summary....186
Table 39. Valid Combinations of ICR Fields....196
Table 40. Power Management Categories....215

Table 41. ACPI-State Support by System Class ....216
Table 42. Required SMAF Code to Stop Grant Mapping ....217
Table 43. Low FID Frequency Table (< 1600 MHz) ....223
Table 44. High FID Frequency Table (>= 1600 MHz) ....224
Table 45. Sample VST Values ....233
Table 46. MVS Values ....234
Table 47. RVO Values ....235
Table 48. VID Code Voltages ....235
Table 49. IRT Values ....236
Table 50. _PSS Status Field ....237
Table 51. Performance State Block Structure ....240
Table 52. Configuration Register Settings for Power Management ....242
Table 53. FADT Table Entries ....243
Table 54. Processor Functional Unit Encoding ....248
Table 55. Performance Monitor Events ....248
Table 56. General MSRs ....261
Table 57. AMD Athlon™ 64 Processor and AMD Opteron™ Processor MSRs ....285
Table 58. FID Code Translations ....299

# Revision History

| Date | Rev. | Description |
|---|---|---|
| July 2003 | 3.04 | Added examples to "DRAM Address Mapping in Interleaving Mode" and "DRAM Address Mapping in Non-Interleaving Mode" sections.<br>Added recommendation for setting "HyperTransport™ FIFO Read Pointer Optimization Register".<br>Added "Spurious Interrupts Caused by Timer Tick Interrupt" section.<br>Added "XSDT Table" section.<br>Added "Detect Target Operating Mode Callback" section.<br>Modified "Multiprocessing Capability Detection" section.<br>Added requirements for AMD Cool'n'Quiet™ technology and BIOS P-state transitions to "Power and Thermal Management" chapter.<br>Added ClkRampHyst (Function 3, Offset D4h) setting to "Power and Thermal Management" chapter.<br>General cleanup in "Power and Thermal Management" chapter.<br>Added "Twtr (Write to Read Delay)" section.<br>Added a requirement to "GART Aperture Base Register".<br>Added a requirement for GartEn (Function 3, Offset 90h) programming.<br>Removed recommandation for extended range temperature sensors in DiodeOffset (Function 3, Offset E4h) definition.<br>Added a requirement for extended configuration space access to "Memory-Mapped I/O Address Map Registers" section. |

| Date | Rev. | Description |
|---|---|---|
| May 2003 | 3.02 | Added "ECC and Chip Kill Error Checking and Correction" section.<br>Clarified "Scrub Control Register" and "DRAM Scrub Address Registers" sections.<br>Clarified LinkFail (Function 0, Offsets 84h, A4h, C4h).<br>Clarified node interleaving in "DRAM Address Map" section.<br>Clarified and corrected DRAM address interleaving in "DRAM CS Base Address Registers" and "DRAM Bank Address Mapping Register" sections.<br>Clarified "tCL (CAS Latency)" section.<br>Added "piggyback scrubbing" description to "Sources of Machine Check Errors" section.<br>Clarified Table 30, "Sources of Machine Check Errors," on page 145.<br>Clarified MCA NB PCI configuration register to MCA MSR mapping in "North-bridge (NB)" section.<br>Corrected MaxLVTEntry (APIC_VER Register, Offset 30h) default value.<br>General cleanup in "Power and Thermal Management" chapter.<br>Added "HyperTransport™ Link Frequency Selection" section. |
| April 2003 | 3.00 | Initial Public release. |

# 1 Introduction and Overview

The *BIOS and Kernel Developer's Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors* is intended for programmers involved in the development of low-level BIOS (basic input/output system) functions, drivers, and operating system kernel modules. It assumes previous experience in microprocessor programming, as well as fundamental knowledge of legacy x86 and AMD64 microprocessor architecture. The reader should also have previous experience in BIOS or OS kernel design issues, as related to microprocessor systems, and a familiarity with various platform technologies, such as DDR, HyperTransport™ technology, and JTAG.

## 1.1 About This Guide

This guide covers the *implementation-specific* features of AMD Athlon™ 64 and AMD Opteron™ processors, as opposed to architectural features. AMD64 architectural features include the AMD64 technology, general-purpose, multimedia, and x87 floating-point registers, as well as other programmer-visible features defined to be constant across all processors. A subset of implementation-specific features are not defined by the processor architectural specifications. These implementation-specific features may differ in various details from one implementation to the next.

***Note:*** *The term* processor *in this document refers to AMD Athlon™ 64 processor architecture and AMD Opteron™ processor architecture. This document covers both classes of devices. For details about differences between them, see the* AMD Athlon™ 64 Processor Data Sheet, *order# 24659 and the* AMD Opteron™ Processor Data Sheet*, order# 23932.*

The implementation-specific features covered in the following chapters include:

- Model-specific registers
- Processor initialization
- Integrated memory system configuration
- HyperTransport technology fabric initialization
- Performance monitoring and special debug features
- DRAM configuration
- Machine check error codes
- Thermal and power management

## 1.2 Related Documents

The references listed in Table 1 may prove invaluable towards a complete understanding of the subject matter in this volume.

**Table 1. Related Documents**

| Title | Order# |
|---|---|
| *AMD64 Architecture Programmer's Manual*, Volume 1, *Application Programming* | 24592 |
| *AMD64 Architecture Programmer's Manual*, Volume 2, *System Programming* | 24593 |
| *AMD64 Architecture Programmer's Manual*, Volume 3, *General-Purpose and System Instructions*<br>*AMD64 Architecture Programmer's Manual*, Volume 4, *128-Bit Media Instructions*<br>*AMD64 Architecture Programmer's Manual*, Volume 5, *64-Bit Media and x87 Floating-Point Instructions* | 24594 (three-volume kit) |
| *AMD Opteron™ Processor Data Sheet* | 23932 |
| *AMD Athlon™ 64 Processor Data Sheet* | 24659 |
| *AMD Processor Recognition Application Note* | 20734 |
| *CPUID Guide for AMD Athlon™ 64 and AMD Opteron™ Processors* | 25481 |
| *Revision Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors* | 25759 |
| *HyperTransport™ I/O Link Specification, Rev. 1.03* | http://www.hypertransport.org |

## 1.3 Conventions and Definitions

Some of the following definitions assume a knowledge of the legacy x86 architecture. See Table 1 for documents that include information on the legacy x86 architecture.

Additional definitions are provided in the glossary at the end of this book, beginning on page 305. That glossary includes the most important terminology of the AMD64 architecture.

### 1.3.1 Notation

**1011b.** A binary value. In this example, a 4-bit value is shown.

**F0EAh.** A hexadecimal value. In this example, a 2-byte value is shown.

**[1,2].** A range that includes the left-most value (in this case, 1) but excludes the right-most value (in this case, 2).

**7–4.** A bit range, from bit 7 to 4, inclusive. The high-order bit is shown first.

**#GP(0).** Notation indicating a general-protection exception (#GP) with error code of 0.

**CR0–CR4.** A register range, from register CR0 through CR4, inclusive, with the low-order register first.

**CR0.PE = 1.** Notation indicating that the PE bit of the CR0 register has a value of 1.

**DS:rSI.** The contents of a memory location whose segment is DS and whose byte address is located in the rSI register.

**EFER.LME = 0.** Notation indicating that the LME bit of the EFER register has a value of 0.

**FF /0.** Notation indicating that FF is the first byte of an opcode and a sub-opcode field in the MODRM byte has a value of 0.

## 1.4 Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors

Some changes to the register set are introduced with different silicon revisions. Refer to the *Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25759 for information about how to identify different processor revisions. The following summarizes register changes after the initial revision.

**Revision C**.

- MSR C001_0043h (ThermTrip_STATUS Register) deleted.
- Function 2, offset 90h, DramEn (bit 10), MemClrStatus (bit 11) added.
- Function 2, offset CCh, DisTscCapture (bit 30) added.
- Function 3, offset 70h, DispRefReq (bits 21-20) added.
- Function 3, offset 74h, DispRefReq (bits 22-20) added.
- MSR C001_001Fh (NB_CFG register), DisDatMsk (bit 36) added.
- Function 3, offset 90h, DisGartTblWlkPrb (bit 6) added.
- Function 3, offset D4h, ClkRampHyst (bits 10-8) added.
- MSR C001_0015 (HWCR register), HLTXSYCEN (bit 12) added.
- MSR C001_0010 (SYSCFG register), ClVicBlkEn (bit 11) deleted.
- MSRs C001_0050h, C001_0051h, C001_0052h, C001_0053h (IOTRAP_ADDR*i* registers), and MSR C001_0054h (IOTRAP_CTL register) added.

# 2 Processor Initialization and Configuration

Each AMD Athlon™ 64 and AMD Opteron™ processor based system has one processor with its HyperTransport™ link connected to a HyperTransport™ I/O hub. When a reset signal is applied to an AMD Athlon™ 64 system or an AMD Opteron™ system, this processor is initialized as the bootstrap processor (BSP). In a multiple processor system, any other processors are initialized as application processors (APs). The BSP begins executing code from the reset vector (0xFFFFFFF0), while each of the APs waits for its Request Disable bit to clear to 0. An AP does not fetch code until its Request Disable bit is cleared. Both BSP and AP operate in 16-bit Real mode after reset. The BSP has the Boot Strap Processor bit set in its APIC_BASE (001Bh) MSR, and each AP has this bit cleared.

The processor node is addressed by its Node ID on the HyperTransport link and can be accessed with a device number in the PCI configuration space on Bus 0. The Node ID 0 is mapped to Device 24, the Node ID 1 is mapped to Device 25, and so on. The BSP is initialized with Node ID 0 after reset, and all APs are initialized with Node ID 7.

The BSP is ready to access its Northbridge and memory controller after its routing table is enabled. The APs are not accessible from the BSP until the links and the routing table are configured.

The initial processor states are described in the "Processor Initialization State" section of the *AMD64 Architecture Programmer's Manual:* Volume 2, *System Programming*.

## 2.1 Bootstrap Processor Initialization

The BSP is responsible for the execution of the BIOS Power-On Self Test (POST) and initialization of APs. The BSP must perform the following tasks in an AMD Athlon™ 64 and AMD Opteron™ processor system:

- AP detection and routing table initialization
- Noncoherent HyperTransport device initialization
- Link width and frequency initialization
- Memory controller initialization on all processor nodes
- Address map table initialization

### 2.1.1 Detecting AP and Initializing Routing Table

The AMD Opteron™ processor has three HyperTransport links; therefore, an AMD Opteron™ system can have from one to eight processors. The BSP must detect all APs in the system, starting from its adjacent processor. Since the APs are initialized at Node 7, the BSP must set entry 7 of its

Routing Table before the AP can be accessed. The adjacent AP will respond to a PCI configuration read if it is present. A new Node ID can be assigned to the AP after the routing information is updated in the routing table. Chapter 8, "HyperTransport™ Technology Configuration and Enumeration" explains the steps for enumerating and initializing routing tables in one-processor (UP), two-processor (DP), and *n* processor systems.

The AMD Athlon™ 64 processor has one HyperTransport link. Because one link must be connected to the HyperTransport I/O hub, an AMD Athlon™ 64 system can only be used in a uniprocessor system.

### 2.1.2 Initializing Noncoherent HyperTransport™ Technology Devices

Once all APs have been detected and the routing tables and link control registers have been initialized, the BSP must initialize noncoherent HyperTransport technology devices.

A noncoherent HyperTransport technology device has either one or two links. A tunnel device has two links and a terminal device has one link. The HyperTransport I/O hub is a typical example of a terminal noncoherent HyperTransport device. The noncoherent HyperTransport device is identified by its Unit ID in a noncoherent HyperTransport link and can be accessed with a device number in the PCI configuration space. The device number in PCI space is the same as the Unit ID assigned to the noncoherent HyperTransport device.

After reset, the Unit ID of each noncoherent HyperTransport technology device is initialized with a value of 0. When a PCI configuration read is performed from the BSP through a noncoherent HyperTransport link, the noncoherent HyperTransport device connected to the port responds. A new non-zero Unit ID value must be assigned to this device. The BIOS continues this process until no device responds at Device 0. The bus number range must be set to the noncoherent HyperTransport link on the BSP and in the address map table prior to the detection. The link control and status registers of a noncoherent device are implemented in the capability register block.

The noncoherent HyperTransport technology devices connected to a port on the AP node can be detected in the same way. Again, the bus number range must be set to the noncoherent HyperTransport link on this AP and in the address map table. The noncoherent device can be detected with the starting bus number, Device 0 on the PCI configuration space.

A noncoherent HyperTransport technology device may use more than one Unit ID. The new ID assigned to a device is its starting ID, the Base Unit ID (BaseUID). The next logic device in this noncoherent HyperTransport device can be identified with BaseUID + 1, and so on. The Unit ID Count field in the Capabilities register indicates how many Unit IDs this device uses.

### 2.1.3 Initializing Link Width and Frequency

The HyperTransport link width and frequency are initialized between the adjacent coherent and/or noncoherent HyperTransport technology devices during the reset sequence. After AP and noncoherent HyperTransport device detection, the link width and frequency can be changed based on the capability of the adjacent devices or the implementation of the system. To make the new link

width and frequency take effect, an LDTSTOP_L needs to be asserted or a warm reset must be performed. The BIOS must guarantee that LDTSTOP_L assertion for link width and frequency changes does not occur within 200µs of reset.

### 2.1.4 Initializing the Memory Controller on All Processor Nodes

The BSP is responsible for configuring the memory controller on all processor nodes and for initializing the DRAM map table. Chapter 4, "DRAM Configuration," describes the functionality of the memory controller and the steps required to initialize memory.

After memory configuration, the memory address MSRs must be set accordingly. The Top of Memory MSR is used for the top of DRAM below 4 Gbytes, and TOP_MEM2 is set to the top of DRAM above 4 Gbytes.

### 2.1.5 Initializing the Address Map Table

Each processor node has an address map table. This table contains the address map for the DRAM area, the address map for PCI memory spaces (MMIO), the address map for PCI I/O spaces, and the bus number range for each noncoherent HyperTransport link in AMD Athlon™ 64 and AMD Opteron™ processors. The address map table must be duplicated on all AMD Athlon™ 64 and AMD Opteron™ processors in a system.

## 2.2 Application Processor Initialization

The application processor (AP) waits until its request disable bit is cleared to 0. The BIOS may clear this bit for an AP a soon as the AP node detection is completed and the routing tables are initialized, or just before the AP register initialization.

When the request disable bit is cleared, the corresponding AP starts to fetch code from the reset vector (0xFFFFFFF0). The BSP bit in AP APIC_BASE register is cleared at reset, so it can be used to terminate the AP execution after the initial APIC initialization.

The AP register initialization is the same as that of other AMD x86 processor families, such as the AMD Athlon™ processors.

# 2.3 BIOS Requirement for 64-Bit Operation

In general, the BIOS need not be aware of 64-bit mode in POST. There are no additional requirements to support 64-bit mode. The operating system determines whether to enable 64-bit mode after the BIOS invokes the operating system loader in legacy mode. There are two areas that need additional explanation:

- Sizing and testing memory above 4 Gbytes
- Using BIOS callback when in 64-bit mode

## 2.3.1 Sizing and Testing Memory above 4 Gbytes

AMD Athlon™ 64 and AMD Opteron™ processors have extended physical address extension (PAE) to support a 40-bit address space. Thus, the BIOS can set up a 32-bit page table that allows it to size and test all physical memory in the system.

## 2.3.2 Using BIOS Callbacks in 64-Bit Mode

A BIOS callback is a mechanism that allows an operating system to call a service routine in the BIOS. BIOS callbacks on an AMD Athlon™ 64 processor or an AMD Opteron™ processor platform in 64-bit mode are not supported by AMD64 operating systems. An operating system loader must be invoked in legacy mode with paging disabled, so that the loader can use 16/32-bit BIOS callbacks.

ACPI code and operating system drivers are not affected by AMD64 operating systems that do not use a BIOS callback in 64-bit mode. ACPI is coded in ASL, which is not affected by the operating system mode. AMD64 operating system drivers are not allowed to use ROM callbacks.

# 3 Memory System Configuration

## 3.1 Configuration Space Accesses

The AMD Athlon™ 64 and AMD Opteron™ processors implement configuration space as defined in the PCI Local Bus Specification, Rev. 2.2, and the HyperTransport™ I/O Link Specification, Rev. 1.03.

Both coherent HyperTransport links and the compatibility bus (the bus to the HyperTransport I/O hub) are accessed through Bus 0. Configuration accesses to Bus 0, devices 0 to 23, are transmitted to the compatibility noncoherent HyperTransport bus. Coherent HyperTransport device configuration space is accessed by using Bus 0, devices 24 to 31, where Device 24 corresponds to Node 0 and Device 31 corresponds to Node 7.

All configuration cycles are type 1 on coherent HyperTransport links. When transmitted down a noncoherent HyperTransport chain by the host bridge, configuration cycles are translated to type 0 if they target the noncoherent HyperTransport bus immediately behind the host bridge. If they target a subordinate bus, they remain as type 1.

Accesses to memory system configuration registers are controlled through the Configuration Address register (see "Configuration Address Register" on page 25) and the Configuration Data register (see "Configuration Data Register" on page 26), which can be accessed through I/O reads/writes to addresses 0CF8h and 0CFCh, respectively.

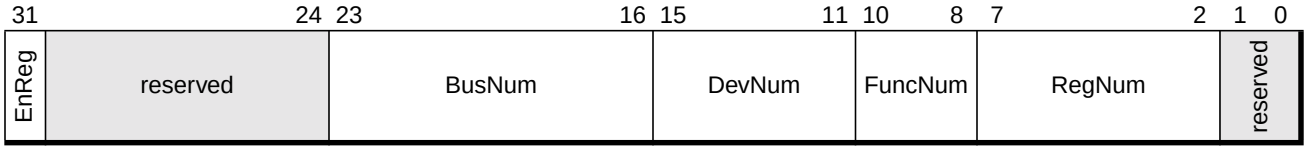
### 3.1.1 Configuration Address Register

Writes to the Configuration Address register specify the target of a configuration access in terms of the bus, device, function, and register. Access to the Configuration Address register must be full doubleword reads or writes.

To access one of the memory system configuration registers defined in this chapter, the bus number should be 0, the device number should be the target AMD Athlon™ 64 processor or AMD Opteron™ processor node number plus 24, and the function and register number should be set based on the register function and offset as specified in this chapter.

When the Enable bit of the Configuration Address register is set, reads and writes to the Configuration Data register access the register specified in the Configuration Address register.

**Configuration Address Register** **0CF8h (doubleword)**



| Bit | Mnemonic | Function | R/W |
|---|---|---|---|
| 31 | EnReg | Enable Register | R/W |
| 30–24 | reserved | | R |
| 23–16 | BusNum | Bus Number | R/W |
| 15–11 | DevNum | Device Number | R/W |
| 10–8 | FuncNum | Function Number | R/W |
| 7–2 | RegNum | Register Number | R/W |
| 1–0 | reserved | | R |

### Field Descriptions

**Register Number (RegNum)**—Bits 7–2. Specifies the doubleword offset of the configuration address.

**Function Number (FuncNum)**—Bits 10–8. Specifies the function number of the configuration address.

**Device Number (DevNum)**—Bits 15–11. Specifies the device number of the configuration address.

**Bus Number (BusNum)**—Bits 23–16. Specifies the bus number of the configuration address.

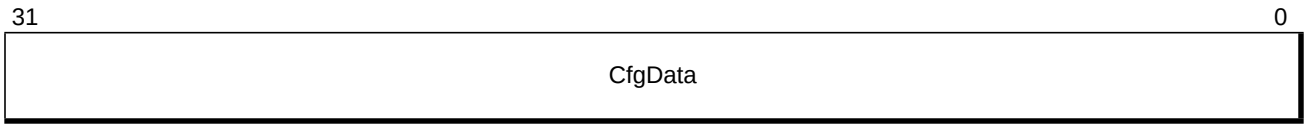
**Enable (EnReg)**—Bit 31. When this bit is set, aligned doubleword accesses to the Configuration Data Register result in configuration-space transactions.

0 = Configuration transactions disabled.
1 = Configuration transactions enabled.

## 3.1.2 Configuration Data Register

Accesses to the Configuration Data Register are translated to configuration-space transactions at the address specified by the Configuration Address Register.

### Configuration Data Register 0CFCh (Doubleword)




| Bits | Mnemonic | Function | R/W |
|---|---|---|---|
| 31–0 | CfgData | Configuration Data | R/W |

### Field Descriptions

**Configuration Data (CfgData)**—Bits 31–0.

## 3.2 Memory System Configuration Registers

The AMD Athlon™ 64 and AMD Opteron™ processors implement memory system configuration registers in the "Northbridge" block of the processor. These registers are mapped into the coherent HyperTransport technology configuration space (Bus 0, device numbers 24–31). The number of devices implemented is equal to the number of processor nodes in the system. Configuration space Device 24 corresponds to Node 0 and is normally the bootstrap processor (BSP). Configuration space device numbers 25 through 31 correspond to nodes 1 through 7, which may or may not be implemented.

The processor implements configuration registers in PCI configuration space using the following four headers:

- Function 0: HyperTransport technology configuration (see page 27)
- Function 1: Address map configuration (see page 53)
- Function 2: DRAM configuration (see page 66)
- Function 3: Miscellaneous configuration (see page 88)

Each of these functions are detailed in the following sections.

***Note:*** *Contents of some fields in the headers and HyperTransport technology capability blocks are maintained through a warm reset. If not specified otherwise, a field is initialized by a warm reset.*

## 3.3 Function 0—HyperTransport™ Technology Configuration

**Table 2. Function 0 Configuration Registers**

| Offset | Register Name | | | | Reset | Access | Description |
|---|---|---|---|---|---|---|---|
| 00h | Device ID | | Vendor ID (AMD) | | 1100_1022h | RO | page 29 |
| 04h | Status | | Command[1] | | 0010_0000h | RO | page 30 |
| 08h | Base Class Code | Subclass Code | Program Interface | Revision ID | 0600_0000h | RO | page 30 |
| 0Ch | BIST | Header Type | Latency Timer | Cache Line Size | 0080_0000h | RO | page 31 |
| 10h | Base Address 0 | | | | 0000_0000h | RO | |

***Notes:***
1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

**Table 2. Function 0 Configuration Registers (Continued)**

| Offset | Register Name | | | | Reset | Access | Description |
|---|---|---|---|---|---|---|---|
| 14h | Base Address 1 | | | | 0000_0000h | RO | |
| 18h | Base Address 2 | | | | 0000_0000h | RO | |
| 1Ch | Base Address 3 | | | | 0000_0000h | RO | |
| 20h | Base Address 4 | | | | 0000_0000h | RO | |
| 24h | Base Address 5 | | | | 0000_0000h | RO | |
| 28h | Card Bus CIS Pointer | | | | 0000_0000h | RO | |
| 2Ch | Sub-System ID | | Sub-System Vendor ID | | 0000_0000h | RO | |
| 30h | ROM Base Address | | | | 0000_0000h | RO | |
| 34h | Capabilities Pointer | | | | page 31 | RO | page 31 |
| 38h | reserved | | | | 0000_0000h | RO | |
| 3Ch | Max Latency | Min GNT | Int Pin | Int Line | 0000_0000h | RO | |
| 40h | Routing Table Node 0 | | | | 0001_0101h | RW | page 32 |
| 44h | Routing Table Node 1 | | | | 0001_0101h | RW | page 32 |
| 48h | Routing Table Node 2 | | | | 0001_0101h | RW | page 32 |
| 4Ch | Routing Table Node 3 | | | | 0001_0101h | RW | page 32 |
| 50h | Routing Table Node 4 | | | | 0001_0101h | RW | page 32 |
| 54h | Routing Table Node 5 | | | | 0001_0101h | RW | page 32 |
| 58h | Routing Table Node 6 | | | | 0001_0101h | RW | page 32 |
| 5Ch | Routing Table Node 7 | | | | 0001_0101h | RW | page 32 |
| 60h | Node ID | | | | 0000_0000h | RW | page 34 |
| 64h | Unit ID | | | | 0000_00E4h | RW | page 35 |
| 68h | HyperTransport™ Transaction Control | | | | 0F00_0000h | RW | page 36 |
| 6Ch | HyperTransport™ Initialization Control | | | | page 40 | RW | page 40 |
| 80h | LDT0 Capabilities | | | | page 42 | RO | page 42 |
| 84h | LDT0 Link Control | | | | 0011_0000h | RW | page 43 |
| 88h | LDT0 Frequency/Revision | | | | page 47 | RW | page 47 |
| 8Ch | LDT0 Feature Capability | | | | 0000_0002h | RO | page 48 |
| 90h | LDT0 Buffer Count | | | | page 49 | RW | page 49 |
| 94h | LDT0 Bus Number | | | | 0000_0000h | RW | page 51 |
| 98h | LDT0 Type | | | | page 51 | RO | page 51 |
| A0h | LDT1 Capabilities | | | | page 42 | RO | page 42 |
| A4h | LDT1 Link Control | | | | 0011_0000h | RW | page 43 |

***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

**Table 2. Function 0 Configuration Registers (Continued)**

| Offset | Register Name | Reset | Access | Description |
|---|---|---|---|---|
| A8h | LDT1 Frequency/Revision | page 47 | RW | page 47 |
| ACh | LDT1 Feature Capability | 0000_0002h | RO | page 48 |
| B0h | LDT1 Buffer Count | page 49 | RW | page 49 |
| B4h | LDT1 Bus Number | 0000_0000h | RW | page 51 |
| B8h | LDT1 Type | page 51 | RO | page 51 |
| C0h | LDT2 Capabilities | page 42 | RO | page 42 |
| C4h | LDT2 Link Control | 0011_0000h | RW | page 43 |
| C8h | LDT2 Frequency/Revision | page 47 | RW | page 47 |
| CCh | LDT2 Feature Capability | 0000_0002h | RO | page 48 |
| D0h | LDT2 Buffer Count | page 49 | RW | page 49 |
| D4h | LDT2 Bus Number | 0000_0000h | RW | page 51 |
| D8h | LDT2 Type | page 51 | RO | page 51 |

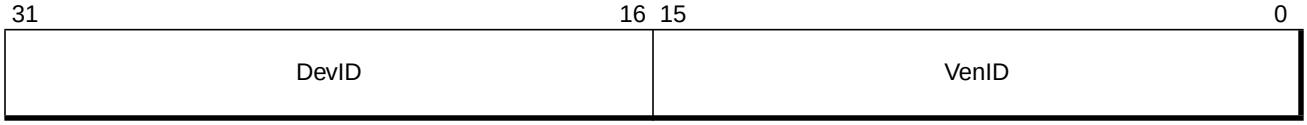
***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

## 3.3.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 0 registers and is part of the standard PCI configuration header.

**Device/Vendor ID Register** **Function 0: Offset 00h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | DevID | Device ID | R | 1100h |
| 15–0 | VenID | Vendor ID | R | 1022h |

### Field Descriptions

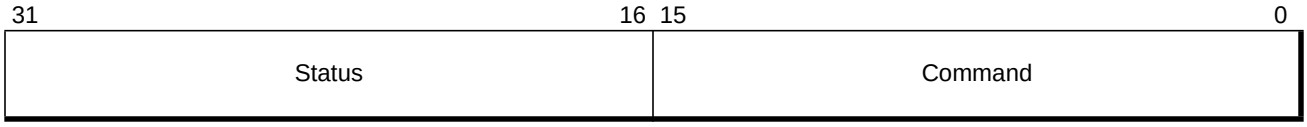
**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1100h for the HyperTransport technology configuration function.

## 3.3.2 Status/Command Register

This register contains status and command information for the Function 0 registers and is part of the standard PCI configuration header.

**Status/Command Register** **Function 0: Offset 04h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | | Status | R | 0010h |
| 15–0 | | Command | R | 0000h |

### Field Descriptions

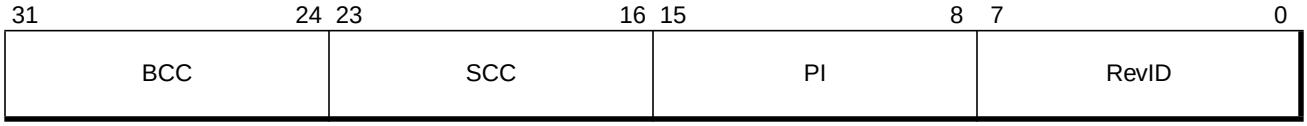
**Command**—Bits 15–0. This read-only value is defined as 0000h.

**Status**—Bits 31–16. This read-only value is defined as 0010h to indicate that the processor has a capabilities list containing configuration information specific to HyperTransport technology.

## 3.3.3 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 0 registers and is part of the standard PCI configuration header.

**Class Code/Revision ID Register** **Function 0: Offset 08h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BCC | Base Class Code | R | 06h |
| 23–16 | SCC | Subclass Code | R | 00h |
| 15–8 | PI | Programming Interface | R | 00h |
| 7–0 | RevID | Revision ID | R | 00h |

### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

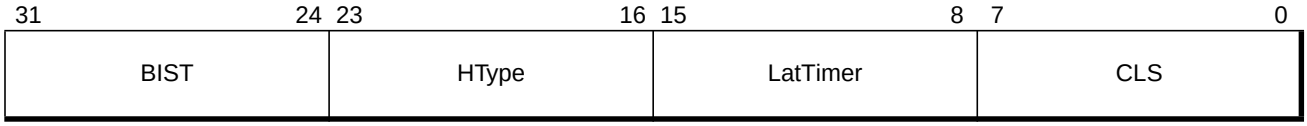
**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

## 3.3.4 Header Type Register

This register specifies the header type for the Function 0 registers and is part of the standard PCI configuration header.

**Header Type Register** **Function 0: Offset 0Ch**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BIST | BIST | R | 00h |
| 23–16 | HType | Header Type | R | 80h |
| 15–8 | LatTimer | Latency Timer | R | 00h |
| 7–0 | CLS | Cache Line Size | R | 00h |

### Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (HType)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

## 3.3.5 Capabilities Pointer Register

This register contains a capabilities pointer to a linked capabilities list of registers specific to HyperTransport technology, with one set for each HyperTransport link. It is part of the standard PCI configuration header.

**Capabilities Pointer Register** **Function 0: Offset 34h**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | R | 0 |
| 7–0 | CapPtr | Capability Pointer | R | |

### Field Descriptions

**Capability Pointer (CapPtr)**—Bits 7–0. Points to the first available HyperTransport technology capabilities block. Depending on the specific HyperTransport link configuration physically present on the processor, this will be either 80h (LDT0), A0h (LDT1) or C0h (LDT2).

## 3.3.6 Routing Table Node *i* Registers

The routing table contains three entries—one for requests RQRoute[7:0], one for responses RPRoute[7:0], and one for broadcasts BCRoute[7:0].

A maximum of eight nodes with three links per node is supported in an AMD Opteron™ system.

These table entries can be read and written, and its contents are not maintained through a warm reset. At reset, all table entries are initialized to the value 01h, indicating that packets should be accepted by this node. Care must be exercised when changing table entries after reset to ensure that connectivity is not lost during the process.

### 3.3.6.1 Request Routing Table

Each node contains a routing table for use with directed requests, with one entry for each of the eight possible destination Node IDs. The value in each entry indicates which outgoing link is used for request packets directed to that particular destination node (DestNode). A 1 in a given bit position indicates that the request is routed through the corresponding output link. Bit 0, when set to 1, indicates that the request must be accepted by this node.

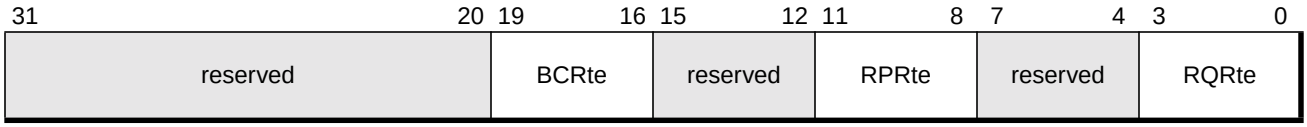
### 3.3.6.2 Response Routing Table

Each node contains a routing table for use with responses, with one entry for each of the eight possible destination node IDs. The value in each entry indicates which outgoing link is used for response packets directed to that particular destination node (DestNode). A 1 in a given bit position indicates that the response is routed through the corresponding output link. Bit 0, when set to 1, indicates that the response must be accepted by this node.

### 3.3.6.3 Broadcast Routing Table

Each node contains a routing table for use with broadcast and probe requests with one entry for each of the eight possible source node IDs. Each entry contains a single bit for each of the outbound links from the node. The packet is forwarded on all links with their corresponding links set to a 1. Bit 0 when set to 1 indicates that the broadcast must be accepted by this node.

## Routing Table Node 0–7 Registers — Function 0: Offset 40h, 44h, 48h, 4Ch, 50h, 54h, 58h, 5Ch




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–20 | reserved | | R | 000h |
| 19–16 | BCRte | Broadcast Route | R/W | 1h |
| 15–12 | reserved | | R | 0h |
| 11–8 | RPRte | Response Route | R/W | 1h |
| 7–4 | reserved | | R | 0h |
| 3–0 | RQRte | Request Route | R/W | 1h |

### Field Descriptions

**Request Route (RQRte)**—Bits 3–0. The Request Route field defines the node or link to which a request packet is forwarded. Request packets are routed to only one destination and the table index is based on the destination Node ID.

Bit [0] = Route to this node
Bit [1] = Route to Link 0
Bit [2] = Route to Link 1
Bit [3] = Route to Link 2

**Response Route (RPRte)**—Bits 11–8. The Response Route field defines the node or link to which a response packet is forwarded. Response packets are routed to only one destination and the table index is based on the destination Node ID.

Bit [0] = Route to this node
Bit [1] = Route to Link 0
Bit [2] = Route to Link 1
Bit [3] = Route to Link 2

**Broadcast Route (BCRte)**—Bits 19–16. The Broadcast Route field defines the node or link(s) to which a broadcast packet is forwarded. Broadcasts may be routed to more than one destination. The Node ID that indexes into the table is the source Node ID.

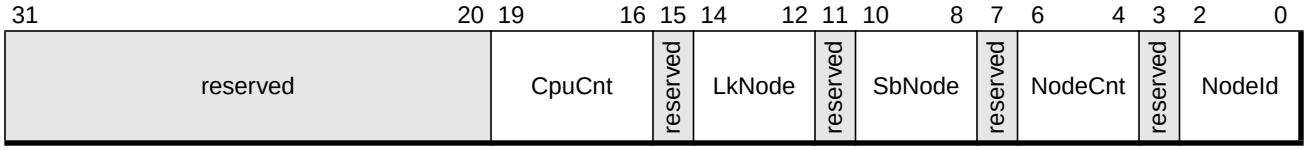
Bit [0] = Route to this node
Bit [1] = Route to Link 0
Bit [2] = Route to Link 1
Bit [3] = Route to Link 2

## 3.3.7 Node ID Register

The Node ID register contains node ID, node count and CPU count information. In general the number of CPUs will be the same as the number of nodes but separate fields are provided in order to allow for nodes which may contain more than one CPU per node.

It is expected that system configuration software will program the Node ID as part of coherent HyperTransport link configuration. Correct system operation depends on an assignment of distinct node ID values not exceeding NodeCnt[2:0]. For example, the Node IDs in a 4-node system must be 0, 1, 2, and 3; an example of an incorrect Node ID assignment in this system is 0, 1, 3, and 4. The Node ID will also be used as the initial APIC ID for the local APIC.

**Node ID Register** **Function 0: Offset 60h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–20 | reserved | | R | 0 |
| 19–16 | CPUCnt | CPU Count | R/W | 0 |
| 15 | reserved | | R | 0 |
| 14–12 | LkNode | Lock Controller Node ID | R/W | 0 |
| 11 | reserved | | R | 0 |
| 10–8 | SbNode | HyperTransport I/O Hub Node ID | R/W | 0 |
| 7 | reserved | | R | 0 |
| 6–4 | NodeCnt | Node Count | R/W | 0 |
| 3 | reserved | | R | 0 |
| 2–0 | NodeID | This Node ID | R/W | |

### Field Descriptions

**Node ID (NodeID)**—Bits 2–0. Defines the Node ID of this node. It resets to 0 for the boot strap processor (BSP) and to 7h for all other nodes.

**Node Count (NodeCnt)**—Bits 6–4. Specifies the number of coherent nodes in the system. Note that the hardware allows only values to be programmed into this field that are consistent with the multiprocessor capabilities of the device, as specified in "Northbridge Capabilities Register" on page 131. This field will not be updated if there is an attempt to write a values that is inconsistent with the specified multiprocessor capabilities.

000b = 1 node
001b = 2 nodes
010b = 3 nodes
011b = 4 nodes
111b = 8 nodes

**HyperTransport I/O Hub Node ID (SbNode)**—Bits 10–8. Defines the Node ID of the node that owns the HyperTransport link to the HyperTransport I/O hub.

**Lock Controller Node ID (LkNode)**—Bits 14–12. Defines the Node ID of the node that contains the Lock Controller. The lock controller node (LkNode) must be the same as the HyperTransport I/O hub node (SbNode).

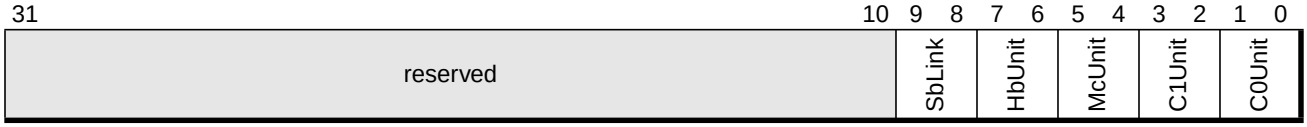
**CPU Count (CPUCnt)**—Bits 19–16. Defines the number of CPUs in the system.

0000b = 1 CPU
1111b = 16 CPUs

## 3.3.8 Unit ID Register

The Unit ID register specifies the Unit ID of each functional unit on the processor. Under normal operation, there is no need for software to change the default Unit ID assignments. This register also contains a pointer to the HyperTransport I/O hub link.

**Unit ID Register** **Function 0: Offset 64h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–10 | reserved | | R | 0 |
| 9–8 | SbLink | HyperTransport I/O Hub Link ID | R/W | 00b |
| 7–6 | HbUnit | Host Bridge Unit ID | R/W | 11b |
| 5–4 | McUnit | Memory Controller Unit ID | R/W | 10b |
| 3–2 | C1Unit | CPU1 Unit ID | R/W | 01b |
| 1–0 | C0Unit | CPU0 Unit ID | R/W | 00b |

### Field Descriptions

**CPU0 Unit ID (C0Unit)**—Bits 1–0. Defines the Unit ID of CPU0.

**CPU1 Unit ID (C1Unit)**—Bits 3–2. Defines the Unit ID of CPU1 (if present).

**Memory Controller Unit ID (McUnit)**—Bits 5–4. Defines the Unit ID of the memory controller.

**Host Bridge Unit ID (HbUnit)**—Bits 7–6. Defines the Unit ID of the host bridge. The host bridge is used to bridge between coherent and noncoherent HyperTransport technology domains.

**HyperTransport I/O Hub Link ID (SbLink)**—Bits 9–8. Defines the link to which the HyperTransport I/O hub is connected. It is only used by the node which owns the HyperTransport I/O hub, as indicated in the Node ID register.

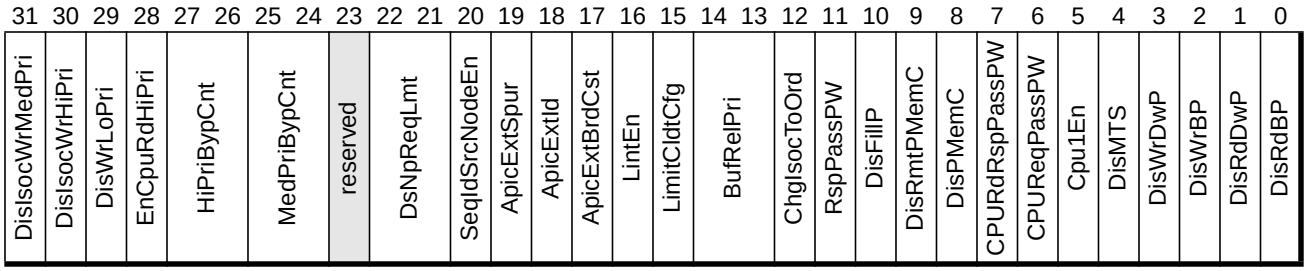
00b = LDT0 (default)
01b = LDT1

10b = LDT2
11b = Undefined

## 3.3.9 HyperTransport™ Transaction Control Register

The HyperTransport Transaction Control register configures the high-level HyperTransport protocols. It can be used to optimize system performance or change HyperTransport technology transaction protocols.

**HyperTransport™ Transaction Control Register** **Function 0: Offset 68h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31 | DisIsocWrMedPri | Disable medium priority isochronous writes | R/W | 0 |
| 30 | DisIsocWrHiPri | Disable high priority isochronous writes | R/W | 0 |
| 29 | DisWrLoPri | Disable low priority writes | R/W | 0 |
| 28 | EnCpuRdHiPri | Enable high priority CPU reads | R/W | 0 |
| 27–26 | HiPriBypCnt | High-priority bypass count | R/W | 11b |
| 25–24 | MedPriBypCnt | Medium-priority bypass count | R/W | 11b |
| 23 | reserved | | R | 0 |
| 22–21 | DsNpReqLmt | Downstream non-posted request limit | R/W | 00b |
| 20 | SeqIdSrcNodeEn | Sequence ID source node enable | R/W | 0 |
| 19 | ApicExtSpur | APIC extended spurious vector enable | R/W | 0 |
| 18 | ApicExtId | APIC extended ID enable | R/W | 0 |
| 17 | ApicExtBrdCst | APIC extended broadcast enable | R/W | 0 |
| 16 | LintEn | Local interrupt conversion enable | R/W | 0 |
| 15 | LimitCldtCfg | Limit coherent HyperTransport configuration space range | R/W | 0 |
| 14–13 | BufRelPri | Buffer release priority select | R/W | 00b |
| 12 | ChgIsocToOrd | Change ISOC to Ordered | R/W | 0 |
| 11 | RspPassPW | Response PassPW | R/W | 0 |
| 10 | DisFillP | Disable fill probe | R/W | 0 |
| 9 | DisRmtPMemC | Disable remote probe memory cancel | R/W | 0 |
| 8 | DisPMemC | Disable probe memory cancel | R/W | 0 |
| 7 | CPURdRspPassPW | CPU Read response PassPW | R/W | 0 |
| 6 | CPUReqPassPW | CPU request PassPW | R/W | 0 |
| 5 | Cpu1En | CPU1 enable | R/W | 0 |
| 4 | DisMTS | Disable memory controller target start | R/W | 0 |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 3 | DisWrDwP | Disable write doubleword probes | R/W | 0 |
| 2 | DisWrBP | Disable write byte probes | R/W | 0 |
| 1 | DisRdDwP | Disable read doubleword probe | R/W | 0 |
| 0 | DisRdBP | Disable read byte probe | R/W | 0 |

## Field Descriptions

**Disable Read Byte Probe (DisRdBP)**—Bit 0. This bit determines if probes are issued for CPU-generated RdSized byte (must be 0 for multiprocessor system; recommended to be 1 for uniprocessor system)

0 = Probes issued
1 = Probes not issued

**Disable Read Doubleword Probe (DisRdDwP)**—Bit 1. This bit determines if probes are issued for CPU-generated RdSized Doubleword (must be 0 for multiprocessor system; recommended to be 1 for uniprocessor system)

0 = Probes issued
1 = Probes not issued

**Disable Write Byte Probes (DisWrBP)**—Bit 2. This bit determines if probes are issued for CPU-generated WrSized byte (must be 0 for multiprocessor system; recommended to be 1 for uniprocessor system)

0 = Probes issued
1 = Probes not issued

**Disable Write Doubleword Probes (DisWrDwP)**—Bit 3. This bit determines if probes are issued for CPU-generated WrSized Doubleword (must be 0 for multiprocessor system; recommended to be 1 for uniprocessor system)

0 = Probes issued
1 = Probes not issued

**Disable Memory Controller Target Start (DisMTS)**—Bit 4. Disables use of TgtStart. TgtStart is used to improve scheduling of back-to-back ordered transactions by indicating when the first transaction is received and ordered at the memory controller.

0 = TgtStart packets are generated
1 = TgtStart packets are not generated

**CPU1 Enable (Cpu1En)**—Bit 5. Enables a second CPU (if present on the node). This bit should only be set for nodes which include two CPUs.

0 = Second CPU disabled or not present
1 = Second CPU enabled

**CPU Request PassPW (CPUReqPassPW)**—Bit 6. Allows CPU-generated requests to pass posted writes.

0 = CPU requests do not pass posted writes
1 = CPU requests pass posted writes

**CPU Read Response PassPW (CPURdRspPassPW)**—Bit 7. Allows RdResponses to CPU-generated reads to pass posted writes.

0 = CPU responses do not pass posted writes
1 = CPU responses pass posted writes

**Disable Probe Memory Cancel (DisPMemC)**—Bit 8. Disables generation of the MemCancel command when a probe hits a dirty cache block. MemCancels are used to attempt to save DRAM and/or HyperTransport technology bandwidth associated with the transfer of stale DRAM data.

0 = Probes may generate MemCancels
1 = Probes may not generate MemCancels

**Disable Remote Probe Memory Cancel (DisRmtPMemC)**—Bit 9. Disables generation of the MemCancel command when a probe hits a dirty cache block unless the probed cache is on the same node as the memory controller. MemCancels are used to attempt to save DRAM and/or HyperTransport technology bandwidth associated with the transfer of stale DRAM data.

0 = Probes hitting dirty blocks generate memory cancel packets, regardless of the location of the probed cache
1 = Only probed caches on the same node as the target memory controller generate memory cancel packets

**Disable Fill Probe (DisFillP)**—Bit 10. Disables probes for CPU-generated fills (must be 0 for multiprocessor system; recommended to be 1 for uniprocessor system).

0 = Probes issued for cache fills
1 = Probes not issued for cache fills

**Response PassPW (RspPassPW)**—Bit 11. Causes the host bridge to set the PassPW bit in all downstream responses.

This technically breaks the PCI ordering rules but it is not expected to be an issue in the downstream direction. Setting this bit improves the latency of upstream requests by allowing the downstream responses to pass posted writes.

0 = Downstream response PassPW based on original request
1 = Downstream response PassPW set to 1

**Change ISOC to Ordered (ChgIsocToOrd)**—Bit 12. Causes bit 1 of RdSz/WrSz commands to be treated as an ordered bit in coherent HyperTransport technology, instead of as an isochronous bit. Setting this bit will disable prioritization of isochronous requests, but will enable tracking of ordered commands across coherent HyperTransport links. This bit should only be set if the performance of non-ordered peer-to-peer traffic across coherent HyperTransport links is optimized, or to disable isochronous prioritization.

0 = Bit 1 of coherent HyperTransport technology sized command used for isochronous prioritization

1 = Bit 1 of coherent HyperTransport technology sized command used for ordering

**Buffer Release Priority Select (BufRelPri)**—Bits 14–13. Selects the number of HyperTransport technology packets sent with a buffer release pending before the buffer release is inserted into the command/data stream of a busy link.

00b = 64 (default)
01b = 16
10b = 8
11b = 2

This should be set to a value of 10 (8 HyperTransport packets) to maximize HyperTransport technology bandwidth.

**Limit Coherent HyperTransport Configuration Space Range (LimitCldtCfg)**—Bit 15. Limits the extent of the coherent HyperTransport technology configuration space based on the number of nodes in the system. When this bit is set, configuration accesses that normally map to coherent HyperTransport space will be sent to noncoherent HyperTransport links instead, if these accesses attempt to access a non-existent node, as specified through the node count in the Node ID register. This bit should be set by BIOS once coherent HyperTransport fabric initialization is complete. Failure to do so will result in PCI configuration accesses to non-existent nodes being sent into the coherent HyperTransport routing fabric, causing the system to hang.

0 = No coherent HyperTransport configuration space restrictions
1 = Limit coherent HyperTransport configuration space based on number of nodes

**Local Interrupt Conversion Enable (LintEn)**—Bit 16. Enables the conversion of broadcast ExtInt/NMI HyperTransport technology interrupts to LINT0/1 before delivering to the local APIC. This conversion only takes place if the local APIC is hardware enabled.

0 = ExtInt/NMI interrupts unaffected
1 = ExtInt/NMI broadcast interrupts converted to LINT0/1

**APIC Extended Broadcast Enable (ApicExtBrdCst)**—Bit 17. Enables extended APIC broadcast functionality.

0 = APIC broadcast is 0Fh
1 = APIC broadcast is FFh

**APIC Extended ID Enable (ApicExtId)**—Bit 18. Enables extended APIC ID functionality.

0 = APIC ID is 4 bits
1 = APIC ID is 8 bits

**APIC Extended Spurious Vector Enable (ApicExtSpur)**—Bit 19. Enables extended APIC spurious vector functionality.

0 = Lower 4 bits of spurious vector are read-only 1111b
1 = Lower 4 bits of spurious vector are writable

**Sequence ID Source Node Enable (SeqIdSrcNodeEn)**—Bit 20. Causes the source node of requests to be used in the SeqID field of downstream noncoherent HyperTransport technology request packets. This may be desirable for some debug HyperTransport packet tracing applications, to match downstream packets with their originating CPU. For normal operation, this bit should be cleared, since this use of SeqID for source node determination may cause problems with packet ordering.

**Downstream non-posted request limit (DsNpReqLmt)**—Bits 22–21. Sets the number of downstream non-posted requests issued by CPU(s) which may be outstanding on the noncoherent HyperTransport links attached to this node. Non-posted requests from CPU(s) will be throttled such that they do not exceed the programmed value in this field.

00b = No limit
01b = Limited to 1
10b = Limited to 4
11b = Limited to 8

**Medium-Priority Bypass Count (MedPriBypCnt)**—Bits 25–24. The maximum number of times a medium priority access can pass a low priority access before medium priority mode is disabled for one access.

**High-Priority Bypass Count (HiPriBypCnt)**—Bits 27–26. The maximum number of times a high priority access can pass a medium or low priority access before high priority mode is disabled for one access.

**Enable High Priority CPU Reads (EnCpuRdHiPri)**—Bit 28. Enables CPU reads to be treated as high priority. CPU reads are treated as medium priority by default. Setting EnCpuRdHiPri changes their priority to high.

**Disable Low Priority Writes (DisWrLoPri)**—Bit 29. Disables non-isochronous writes from being treated as low priority. Non-isochronous writes are treated as low priority by default. Setting DisWrLoPri changes their priority to medium.

**Disable High Priority Isochronous Writes (DisIsocWrHiPri)**—Bit 30. Disables isochronous writes from being treated as high priority. Isochronous writes are treated as high priority by default. Setting DisIsocWrHiPri changes their priority to medium (if DisIsocWrMedPri is clear) or low (if DisIsocWrMedPri is set).

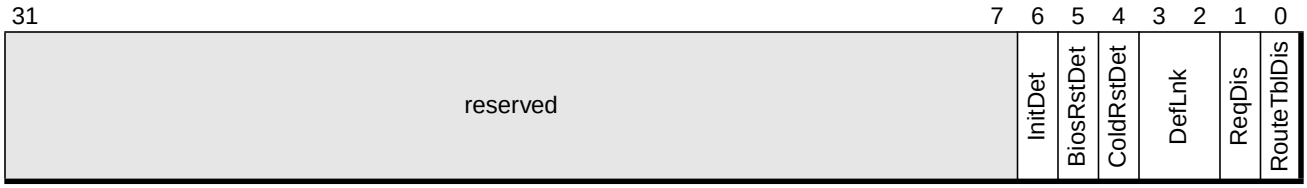
**Disable Medium Priority Isochronous Writes (DisIsocWrMedPri)**—Bit 31. Disables isochronous writes from being treated as medium priority. Isochronous writes are treated as high priority by default. Setting DisIsocWrMedPri along with DisIsocWrHiPri changes their priority to low.

### 3.3.10 HyperTransport™ Initialization Control Register

The HyperTransport Initialization Control register controls the routing and request generation that follows device initialization. This register also contains a set of scratchpad read/write bits that are cleared by different initialization conditions and that may be used to distinguish between various

types of initialization events. Note that the Request Disable and Routing Table Disable bits must not be cleared until the routing tables have been initialized.

### HyperTransport™ Initialization Control Register — Function 0: Offset 6Ch




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–7 | reserved | | R | 0 |
| 6 | InitDet | INIT Detect | R/W | 0 |
| 5 | BiosRstDet | BIOS Reset Detect | R/W | 0 |
| 4 | ColdRstDet | Cold Reset Detect | R/W | 0 |
| 3–2 | DefLnk | Default Link | R | |
| 1 | ReqDis | Request Disable | R/W | |
| 0 | RouteTblDis | Routing Table Disable | R/W | 1 |

### Field Descriptions

**Routing Table Disable (RouteTblDis)**—Bit 0. This bit determines whether the routing tables are used or whether default configuration access routing is used. It resets to 1, thereby routing requests to the Configuration Special Register (CSR) block and routing responses based on DefLnk. Once the routing tables have been set up this bit should be cleared.

0 = Packets are routed according to the routing tables
1 = Packets are routed according to the default link field (DefLnk)

**Request Disable (ReqDis)**—Bit 1. This bit determines if the node is allowed to generate request packets. It resets to 0 for the BSP and to 1 for all other processors. This bit should be cleared by system initialization firmware once the system has been initialized from the BSP. This bit is set by hardware and cleared by software.

0 = Request packets may be generated
1 = Request packets may not be generated

**Default Link (DefLnk)**—Bits 3–2. This field is used after a reset and before the routing tables are initialized. It is used by hardware to determine which link to route responses to and may be used by software as part of system connectivity discovery.

The default link field is loaded each time an incoming request is received, with the link ID of the link on which the packet arrived. It is read-only from software. It is only used to route packets during initialization, when the Routing Table Disable bit is set to 1, and only one outstanding request is active in the system at a time. During this interval, the value in the Default Link field is used to route responses, so that responses are always sent out on the link from which the last request was received. The register is updated as soon as a request is

received, so reads of this register from the fabric return the link ID of the link on which the read command arrived. Reads from the CPU on the local node cause this field to get loaded with 11b.

00b = LDT0
01b = LDT1
10b = LDT2
11b = CPU on same node

**Cold Reset Detect (ColdRstDet)**—Bit 4. This bit may be used to distinguish between a cold versus a warm reset event by setting the bit to 1 before an initialization event is generated. This bit is cleared by a cold reset but not by a warm reset.

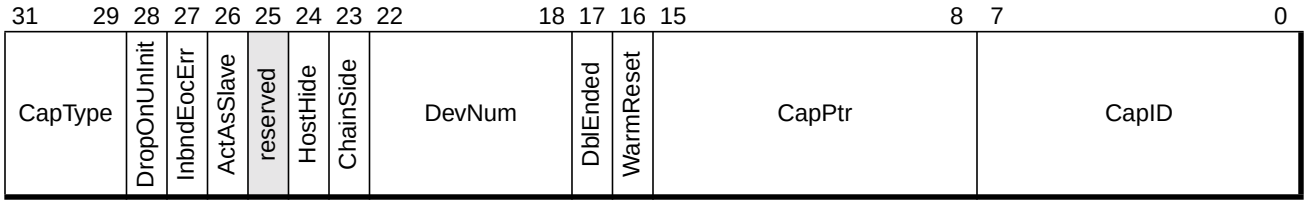
**BIOS Reset Detect (BiosRstDet)**—Bit 5. This bit may be used to distinguish between a reset event generated by the BIOS versus a reset event generated for any other reason by setting the bit to 1 before initiating a BIOS-generated reset event. This bit is cleared by a cold reset but not by a warm reset.

**INIT Detect (InitDet)**—Bit 6. This bit may be used to distinguish between an INIT and a warm/cold reset by setting the bit to 1 before an initialization event is generated. This bit is cleared by a warm or cold reset but not by an INIT.

## 3.3.11 LDT*i* Capabilities Register

These registers define the capabilities of the LDT links. They also contain a capabilities pointer that points to the next item in the linked capabilities list.

**LDT0, LDT1, LDT2 Capabilities Registers** **Function 0: Offset 80h, A0h, C0h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–29 | CapType | Capability Type | R | 001b |
| 28 | DropOnUnInit | Drop on Uninitialized Link | R | 0 |
| 27 | InbndEocErr | Inbound End-of-Chain Error | R | 0 |
| 26 | ActAsSlave | Act As Slave | R | 0 |
| 25 | reserved | | R | 0 |
| 24 | HostHide | Host Hide | R | 1 |
| 23 | ChainSide | Chain Side | R | 0 |
| 22–18 | DevNum | Device Number | R | 0 |
| 17 | DblEnded | Double Ended | R | 0 |
| 16 | WarmReset | Warm Reset | R | 1 |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 15–8 | CapPtr | Capability Pointer | R | |
| 7–0 | CapID | Capability ID (Always Reads 08h) | R | 08h |

### Field Descriptions

**Capability ID (CapID)**—Bits 7–0. A capability ID of 08h indicates HyperTransport technology capability.

**Capability Pointer (CapPtr)**—Bits 15–8. Contains a pointer to the next capability in the list. Depending on the specific HyperTransport link configuration physically present on the processor, this will be either A0h (LDT1), C0h (LDT2), or 00h, if it is the last one.

**Warm Reset (WarmReset)**—Bit 16. Allows a reset sequence initiated by the HyperTransport technology control register to be either warm or cold. Since resets initiated through the HyperTransport control register are not supported, this field is read-only 1.

**Double Ended (DblEnded)**—Bit 17. Indicates that there is another bridge at the far end of the noncoherent HyperTransport chain. Since double-hosted chains are not supported, this field is read-only 0.

**Device Number (DevNum)**—Bits 22–18. The device number of configuration accesses which Northbridge responds to when accessed from the noncoherent HyperTransport technology chain. Since double-hosted chains are not supported, this field is read-only 0.

**Chain Side (ChainSide)**—Bit 23. This bit indicates which side of the host bridge is being accessed. Since double-hosted chains are not supported, this bit is read-only 0.

**Host Hide (HostHide)**—Bit 24. This bit causes the memory system configuration space to be inaccessible from the noncoherent HyperTransport technology chain. Since double-hosted chains are not supported, this bit is read-only 1.

**Act As Slave (ActAsSlave)**—Bit 26. Since this function is not currently implemented, this bit is read-only 0.

**Inbound End-of-Chain Error (InbndEocErr)**—Bit 27. Since this function is not currently implemented, this bit is read-only 0.

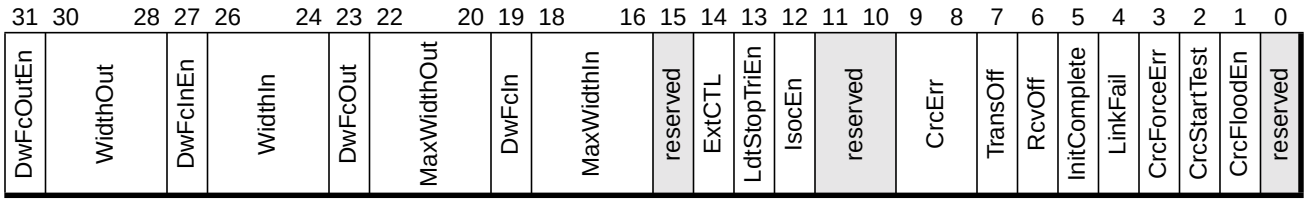
**Drop on Uninitialized Link (DropOnUnInit)**—Bit 28. Since this function is not currently implemented, this bit is read-only 0.

**Capability Type (CapType)**—Bits 31–29. The code 001b designates a host interface HyperTransport technology capability.

## 3.3.12 LDT*i* Link Control Registers

These registers control LDT link operation and log link errors.

## LDT0, LDT1, LDT2 Link Control Registers — Function 0: Offset 84h, A4h, C4h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31 | DwFcOutEn | Doubleword Flow Control Out Enable | R | 0 |
| 30–28 | WidthOut | Link Width Out | R/W | 0 |
| 27 | DwFcInEn | Doubleword Flow Control In Enable | R | 0 |
| 26–24 | WidthIn | Link Width In | R/W | 0 |
| 23 | DwFcOut | Doubleword Flow Control Out | R | 0 |
| 22–20 | MaxWidthOut | Max Link Width Out | R | 001b |
| 19 | DwFcIn | Doubleword Flow Control In | R | 0 |
| 18–16 | MaxWidthIn | Max Link Width In | R | 001b |
| 15 | reserved | | R | 0 |
| 14 | ExtCTL | Extended CTL Time | R/W | 0 |
| 13 | LdtStopTriEn | HyperTransport Stop Tristate Enable | R/W | 0 |
| 12 | IsocEn | Isochronous Enable | R | 0 |
| 11–10 | reserved | | R | 0 |
| 9–8 | CrcErr | CRC_Error | R/W | 0 |
| 7 | TransOff | Transmitter Off | R/W | 0 |
| 6 | RcvOff | Receiver Off | R/W | 0 |
| 5 | InitComplete | Initialization Complete | R | 0 |
| 4 | LinkFail | Link Failure | R/W | 0 |
| 3 | CrcForceErr | CRC Force Error | R/W | 0 |
| 2 | CrcStartTest | CRC Start Test | R | 0 |
| 1 | CrcFloodEn | CRC Flood Enable | R/W | 0 |
| 0 | reserved | | R | 0 |

### Field Descriptions

**CRC Flood Enable (CrcFloodEn)**—Bit 1. If cleared to 0, this bit prevents CRC errors both from generating sync packets and causing the system to come down, and from setting the LinkFail bit. However, CRC checking logic still runs on all lanes enabled by LinkWidthIn, and detected errors still set the CRC Error bits to 1. Its reset value is 0.

0 = Do not generate sync packets on CRC error (default)
1 = Generate sync packets on CRC error

**CRC Start Test (CrcStartTest)**—Bit 2. Since this function is not currently implemented, this bit is read-only 0.

**CRC Force Error (CrcForceErr)**—Bit 3. When this bit is set, a bad CRC is generated on all transmitting lanes as enabled by LinkWidthOut. The covered data is not affected. This bit is readable and writable by software and resets to 0.

0 = Do not generate a bad CRC (default)
1 = Generate a bad CRC

**Link Failure (LinkFail)**—Bit 4. The LinkFail bit is set to 1 when a CRC error or a sync packet is detected after link initialization or if the link is not connected. See "Machine Check Architecture (MCA) Registers" on page 92 for more details. This bit is maintained through a warm reset and is cleared to 0 on a cold reset. LinkFail is set to 1 by hardware in the event of a link error that results in a sync flood. It can be cleared by software by writing a 1.

0 = No link failure detected
1 = Link failure detected

**Initialization Complete (InitComplete)**—Bit 5. This read-only bit is reset to 0, and set to 1 by hardware when low-level link initialization is successfully complete. If there is no device on the other end of the link, or if that device is unable to properly perform the low-level link initialization protocol, the bit is not set to 1. Software must not attempt to generate any packets across the link until this bit is set to 1. CRC checking in the hardware does not begin until initialization is complete.

0 = Initialization not complete
1 = Initialization is complete

**Receiver Off (RcvOff)**—Bit 6. This bit disables the receiver from accepting any further packets. This bit resets to 0, and is set by software writing a 1 to the bit. This bit cannot be cleared by software—a write of 0 to this bit position has no effect. If the HyperTransport link is not connected, then this bit is set by hardware.

0 = Receiver on
1 = Receiver off

**Transmitter Off (TransOff)**—Bit 7. This bit provides a mechanism to shut off a link transmitter for power savings or EMI reduction. When set to 1, no output signals toggle on the link. This bit resets to 0, and is set by software writing a 1 to the bit. This bit cannot be cleared by software—a write of 0 to this bit position has no effect. If the HyperTransport link is not connected, then this bit is set by hardware.

0 = Transmitter on
1 = Transmitter off

**CRC_Error (CrcErr)**—Bits 9–8. These bits are set to 1 by hardware when a CRC error is detected on an incoming link. Errors are detected and reported on a per byte lane basis where bit 8 corresponds to the least significant byte lane. Two bits are required to cover the maximum HyperTransport link width of 16 bits. Error bits for unimplemented (as specified by MaxWidthIn) or unused byte lanes return 0 when read.

These bits are maintained through a warm reset and is cleared to 0 on a cold reset. They are readable from software and are individually cleared to 0 by software by writing a 1.

00b = No error
Bit [0] =1 Error on Byte Lane 0
Bit [1] =1 Error on Byte Lane 1

**Isochronous Enable (IsocEn)**—Bit 12. Since this function is not currently implemented, this bit is read-only 0.

**HyperTransport Stop Tristate Enable (LdtStopTriEn)**—Bit 13. This bit controls whether the transmitter tristates the link during the disconnected state of an LDTSTOP_L sequence. When the bit is set the transmitter tristates the link. When the bit is clear, it continues to drive the link. The bit is cleared by a cold reset and its value is maintained through a warm reset.

0 = Driven during disconnect of an LDTSTOP_L
1 = Tristated during disconnect of an LDTSTOP_L

**Extended CTL Time (ExtCTL)**—Bit 14. When this bit is set, CTL will be asserted for 50µs instead of 16 bit times during the link initialization sequence. This bit should be set if extended CTL is required by the device at the other end of the HyperTransport link, as indicated by the Extended CTL Required bit in the Feature Capability register. The bit is cleared by cold reset and its value is maintained through warm reset.

**Max Link Width In (MaxWidthIn)**—Bits 18–16. This field contains three bits that indicate the physical width of the incoming side of the HyperTransport link implemented by this device. For the AMD Athlon™ 64 and AMD Opteron™ processors, this field is set to 000 or 001 to indicate an 8-bit or 16-bit link.

**Doubleword Flow Control In (DwFcIn)**—Bit 19. This bit is read-only 0 to indicate that this link does not support doubleword flow control.

**Max Link Width Out (MaxWidthOut)**—Bits 22–20. This field contains three bits that indicate the physical width of the outgoing side of the HyperTransport link implemented by this device. It is read-only. For the AMD Athlon™ 64 and AMD Opteron™ processors, this field is set to 000 or 001 to indicate an 8-bit or 16-bit link.

**Doubleword Flow Control Out (DwFcOut)**—Bit 23. This bit is read-only 0 to indicate that this link does not support doubleword flow control.

**Link Width In (WidthIn)**—Bits 26–24. This field is similar to the WidthOut field, except that it controls the utilized width of the incoming side of the links implemented by this device.

The hardware will only allow values to be programmed into this field which are consistent with the width capabilities of the link as specified by MaxWidthIn. Attempts to write values inconsistent with the capabilities will result in this field not being updated.

**Doubleword Flow Control In Enable (DwFcInEn)**—Bit 27. Since this function is not currently implemented, this bit is read-only 0.

**Link Width Out (WidthOut)**—Bits 30–28. This field controls the utilized width (which may not exceed the physical width) of the outgoing side of the HyperTransport link. Software can read/write this field, and its value is maintained through a warm reset. After cold reset, this field is

initialized by hardware based on the results of the link width negotiation sequence described in the HyperTransport technology specification. Based on sizing the devices at both ends of the link, software may then write a different value into the register. The chain must pass through warm reset or a LDTSTOP_L disconnect sequence for the new width values to be reflected on the link.

The WidthOut CSR in the link transmitter must match the WidthIn CSR in the link receiver of the device on the other side of the link. The LinkWidthIn and LinkWidthOut registers within the same device are not required to have matching values.

The hardware will only allow values to be programmed into this field which are consistent with the width capabilities of the link as specified by MaxWidthOut. Attempts to write values inconsistent with the capabilities will result in this field not being updated

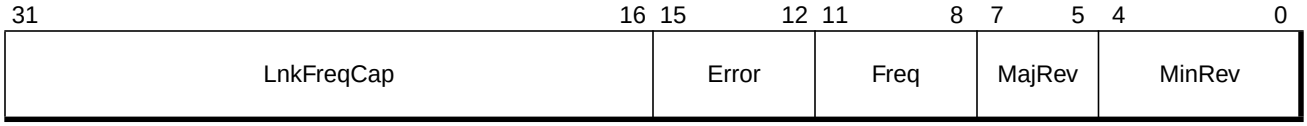
000b = 8-bit
001b = 16-bit
010b = reserved
011b = 32-bit
100b = 2-bit
101b = 4-bit
110b = reserved
111b = Link physically not connected

**Doubleword Flow Control Out Enable (DwFcOutEn)**—Bit 31. Since this function is not currently implemented, this bit is read-only 0.

## 3.3.13 LDT*i* Frequency/Revision Registers

These registers control the link frequency, specify the link frequency capabilities, and describe the HyperTransport technology specification level to which the specific link conforms.

### LDT0, LDT1, LDT2 Frequency/Revision Registers Function 0: Offset 88h, A8h, C8h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | LnkFreqCap | Link Frequency Capability | R | |
| 15–12 | Error | Error | R | 0 |
| 11–8 | Freq | Link Frequency | R/W | 0 |
| 7–5 | MajRev | Major Revision | R | 001b |
| 4–0 | MinRev | Minor Revision | R | 00010b |

### Field Descriptions

**Minor Revision (MinRev)**—Bits 4–0. Contains the minor revision of the HyperTransport technology specification to which the device conforms.

**Major Revision (MajRev)**—Bits 7–5. Contains the major revision of the HyperTransport technology specification level to which the device conforms.

**Link Frequency (Freq)**—Bits 11–8. Specifies the maximum operating frequency of the link's transmitter clock. The encoding of this field is shown below.

The LinkFreq is cleared by cold reset. Software can write a nonzero value to this register, and that value takes effect on the next warm reset or LDTSTOP_L disconnect sequence.

The hardware will only allow values to be programmed into this field which are consistent with the frequency capabilities of the link as specified by LinkFreqCap. Attempts to write values inconsistent with the capabilities will result in this field not being updated.

It is possible to program this field for a higher frequency than the maximum allowed by the processor. Refer to the processor data sheet for the maximum operating frequency allowed for a given processor implementation.

0000b = 200 MHz
0001b = reserved
0010b = 400 MHz
0011b = reserved
0100b = 600 MHz
0101b = 800 MHz
0110b = 1000 MHz
0111b = reserved
1000–1110b = reserved
1111b = 100 MHz

If a link is not connected, then this field should not be written and should be left in its default state.

**Error (Error)**—Bits 15–12. This function is not currently implemented.

**Link Frequency Capability (LnkFreqCap)**—Bits 31–16. Indicates the clock frequency capabilities of the link. Each bit corresponds to one of the 16 possible link frequency encodings.

While the frequency capabilities of different AMD Athlon™ 64 and AMD Opteron™ processors may vary, the 200-MHz and 100-MHz frequencies are always supported.

## 3.3.14 LDT*i* Feature Capability Registers

These registers identify features that are supported by the specific link.

### LDT0, LDT1, LDT2 Feature Capability Registers — Function 0: Offset 8Ch, ACh, CCh




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–9 | reserved | | R | 0 |
| 8 | ExtRegSet | Extended Register Set | R | 0 |
| 7–4 | reserved | | R | 0 |
| 3 | ExtCTLRqd | Extended CTL Required | R | 0 |
| 2 | CrcTstMode | CRC Test Mode | R | 0 |
| 1 | LdtStopMode | HyperTransport Stop Mode | R | 1 |
| 0 | IsocMode | Isochronous Flow Control Mode | R | 0 |

#### Field Descriptions

**Isochronous Flow Control Mode (IsocMode)**—Bit 0. This is a read-only bit that reflects whether isochronous flow control operation is supported. This bit is set to 0, indicating no support for this feature.

**HyperTransport Stop Mode (LdtStopMode)**—Bit 1. This is a read-only bit that indicates whether the LDTSTOP_L protocol is supported. This bit is set to 1, indicating support for this feature.

**CRC Test Mode (CrcTstMode)**—Bit 2. This is a read-only bit that indicates whether CRC test mode is supported. This bit is set to 0, indicating no support for this feature.

**Extended CTL Required (ExtCTLRqd)**—Bit 3. This is a read-only bit that indicates whether extended CTL is required. This bit is set to 0, indicating that extended CTL is not required.

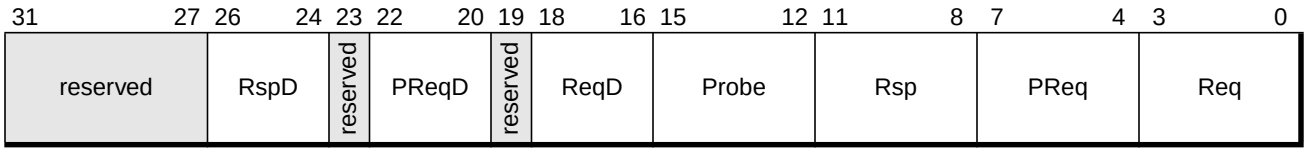
**Extended Register Set (ExtRegSet)**—Bit 8. This is a read-only bit that indicates whether the Enumeration Scratchpad, Error Handling and Memory Base/Limit Upper registers are supported. This bit is set to 0, indicating no support for this feature.

## 3.3.15 LDT*i* Buffer Count Registers

These registers specify the number of command and data buffers for each virtual channel available for use by the transmitter at the other end of the specific link. See "XBAR Flow Control Buffers" on page 113 for more information on command and data buffers.

***Note:*** *The reset values for each of the LDTn Buffer Count registers depend on the link connection type (coherent HyperTransport or noncoherent HyperTransport technology). Because hardware attempts to choose optimal settings, this register should not, in general, need to be changed.*

## LDT0, LDT1, LDT2 Buffer Count Registers — Function 0: Offset 90h, B0h, D0h




| Bits | Mnemonic | Function | R/W | Coherent HyperTransport Reset | Noncoherent HyperTransport Reset |
|---|---|---|---|---|---|
| 31–27 | reserved | | R | 0 | 0 |
| 26–24 | RspD | Response Data Buffer Count | R/W | 100b | 010b |
| 23 | reserved | | R | 0 | 0 |
| 22–20 | PReqD | Posted Request Data Buffer Count | R/W | 001b | 101b |
| 19 | reserved | | R | 0 | 0 |
| 18–16 | ReqD | Request Data Buffer Count | R/W | 011b | 001b |
| 15–12 | Probe | Probe Buffer Count | R/W | 5h | 0h |
| 11–8 | Rsp | Response Buffer Count | R/W | 6h | 4h |
| 7–4 | PReq | Posted Request Buffer Count | R/W | 1h | 5h |
| 3–0 | Req | Request Buffer Count | R/W | 3h | 6h |

### Field Descriptions

**Request Buffer Count (Req)**—Bits 3–0. Defines the number of request command buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Posted Request Buffer Count (PReq)**—Bits 7–4. Defines the number of Posted request command buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Response Buffer Count (Rsp)**—Bits 11–8. Defines the number of response buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Probe Buffer Count (Probe)**—Bits 15–12. Defines the number of probe buffers available for use by the transmitter at the other end of the link. This field must be 0 for a noncoherent HyperTransport link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Request Data Buffer Count (ReqD)**—Bits 18–16. Defines the number of request data buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Posted Request Data Buffer Count (PReqD)**—Bits 22–20. Defines the number of posted request data buffers available for use by the transmitter at the other end of the link. See the register

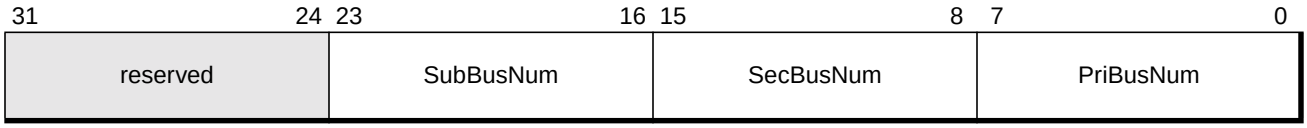
layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Response Data Buffer Count (RspD)**—Bits 26–24. Defines the number of response data buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

## 3.3.16 LDT*i* Bus Number Registers

If the specific link (LDT0, LDT1, or LDT2) is noncoherent HyperTransport technology, this register specifies the bus numbers downstream (behind) the host bridge. If the link is coherent HyperTransport technology, this register has no meaning.

**LDT0, LDT1, LDT2 Bus Number Registers** **Function 0: Offset 94h, B4h, D4h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | reserved | | R | 0 |
| 23–16 | SubBusNum | Subordinate Bus Number | R/W | 0 |
| 15–8 | SecBusNum | Secondary Bus Number | R/W | 0 |
| 7–0 | PriBusNum | Primary Bus Number | R | 0 |

### Field Descriptions

**Primary Bus Number (PriBusNum)**—Bits 7–0. Defines the primary bus number. Because the primary bus is the coherent HyperTransport technology fabric, this field always reads 0.

**Secondary Bus Number (SecBusNum)**—Bits 15–8. Defines the secondary bus number.

The Secondary Bus Number register is used to record the bus number of the bus segment to which the secondary interface of the host bridge is connected. Configuration software programs the value in this register. If this link contains the HyperTransport I/O hub, the secondary bus number must be programmed to 0.

**Subordinate Bus Number (SubBusNum)**—Bits 23–16. Defines the subordinate bus number.

The Subordinate Bus Number register is used to record the bus number of the highest numbered bus segment that is behind (or subordinate to) the host bridge. Configuration software programs the value in this register.

## 3.3.17 LDT*i* Type Registers

These registers designate the type of HyperTransport link attached to the specific link. The bits are set by hardware after link initialization.

## LDT0, LDT1, LDT2 Type Registers — Function 0: Offset 98h, B8h, D8h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–5 | reserved | | R | 0 |
| 4 | LinkConPend | Link Connect Pending | R | |
| 3 | UniP-cLDT | UniP-cLDT | R | |
| 2 | NC | Non Coherent | R | |
| 1 | InitComplete | Initialization Complete | R | |
| 0 | LinkCon | Link Connected | R | |

### Field Descriptions

**Link Connected (LinkCon)**—Bit 0. Indicates that the link is connected. It is valid once the LinkConPend bit is clear.

0 = Not connected
1 = Connected

**Initialization Complete (InitComplete)**—Bit 1. Set to 1 to indicate that the initialization of the link has completed. (It is a duplicate of Bit 5 in HyperTransport Link Control). The NC and UniP-cLDT bits are invalid until link initialization is complete.

0 = Initialization not complete
1 = Initialization is complete.

**Non Coherent (NC)**—Bit 2. Defines the link type (coherent versus noncoherent).

0 = Coherent HyperTransport technology
1 = Noncoherent HyperTransport technology

**UniP-cLDT (UniP-cLDT)**—Bit 3. Further qualifies the NC link type bit. A 1 indicates that this link is a uniprocessor coherent HyperTransport link connected to an external Northbridge.

0 = Normal coherent HyperTransport or noncoherent HyperTransport link
1 = Uniprocessor coherent HyperTransport link to external Northbridge

**Link Connect Pending (LinkConPend)**—Bit 4. Qualifies the LinkCon bit and is set to 1 when hardware is attempting to determine whether a link is connected or not.

0 = Link connection determination complete
1 = Link connection still being determined

# 3.4 Function 1—Address Map

The address map defines the address spaces assigned to DRAM, memory-mapped I/O, PCI I/O, and configuration accesses and also specifies destination information for each to facilitate routing of each access to the appropriate target. Table 3 lists each Function 1 configuration register.

**Table 3. Function 1 Configuration Registers**

| Offset | Register Name | | | | Reset | Access | Description |
|---|---|---|---|---|---|---|---|
| 00h | Device ID | | Vendor ID (AMD) | | 1101_1022h | RO | page 29 |
| 04h | Status[1] | | Command | | 0000_0000h | RO | |
| 08h | Base Class Code | Subclass Code | Program Interface | Revision ID | 0600_0000h | RO | page 55 |
| 0Ch | BIST | Header Type | Latency Timer | Cache Line Size | 0080_0000 | RO | page 56 |
| 10h | Base Address 0 | | | | 0000_0000h | RO | |
| 14h | Base Address 1 | | | | 0000_0000h | RO | |
| 18h | Base Address 2 | | | | 0000_0000h | RO | |
| 1Ch | Base Address 3 | | | | 0000_0000h | RO | |
| 20h | Base Address 4 | | | | 0000_0000h | RO | |
| 24h | Base Address 5 | | | | 0000_0000h | RO | |
| 28h | Card Bus CIS Pointer | | | | 0000_0000h | RO | |
| 2Ch | Sub-System ID | | Sub-System Vendor ID | | 0000_0000h | RO | |
| 30h | ROM Base Address | | | | 0000_0000h | RO | |
| 34h | Capabilities | | | | 0000_0000h | RO | |
| 38h | reserved | | | | 0000_0000h | RO | |
| 3Ch | Max Latency | Min GNT | Int Pin | Int Line | 0000_0000h | RO | |
| 40h | DRAM Base 0 | | | | | RW | page 57 |
| 44h | DRAM Limit 0 | | | | | RW | page 58 |
| 48h | DRAM Base 1 | | | | | RW | page 57 |
| 4Ch | DRAM Limit 1 | | | | | RW | page 58 |
| 50h | DRAM Base 2 | | | | | RW | page 57 |
| 54h | DRAM Limit 2 | | | | | RW | page 58 |
| 58h | DRAM Base 3 | | | | | RW | page 57 |
| 5Ch | DRAM Limit 3 | | | | | RW | page 58 |

***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

**Table 3. Function 1 Configuration Registers (Continued)**

| Offset | Register Name | Reset | Access | Description |
|---|---|---|---|---|
| 60h | DRAM Base 4 | | RW | page 57 |
| 64h | DRAM Limit 4 | | RW | page 58 |
| 68h | DRAM Base 5 | | RW | page 57 |
| 6Ch | DRAM Limit 5 | | RW | page 58 |
| 70h | DRAM Base 6 | | RW | page 57 |
| 74h | DRAM Limit 6 | | RW | page 58 |
| 78h | DRAM Base 7 | | RW | page 57 |
| 7Ch | DRAM Limit 7 | | RW | page 58 |
| 80h | Memory-Mapped I/O Base 0 | | RW | page 60 |
| 84h | Memory-Mapped I/O Limit 0 | | RW | page 61 |
| 88h | Memory-Mapped I/O Base 1 | | RW | page 60 |
| 8Ch | Memory-Mapped I/O Limit 1 | | RW | page 61 |
| 90h | Memory-Mapped I/O Base 2 | | RW | page 60 |
| 94h | Memory-Mapped I/O Limit 2 | | RW | page 61 |
| 98h | Memory-Mapped I/O Base 3 | | RW | page 60 |
| 9Ch | Memory-Mapped I/O Limit 3 | | RW | page 61 |
| A0h | Memory-Mapped I/O Base 4 | | RW | page 60 |
| A4h | Memory-Mapped I/O Limit 4 | | RW | page 61 |
| A8h | Memory-Mapped I/O Base 5 | | RW | page 60 |
| ACh | Memory-Mapped I/O Limit 5 | | RW | page 61 |
| B0h | Memory-Mapped I/O Base 6 | | RW | page 60 |
| B4h | Memory-Mapped I/O Limit 6 | | RW | page 61 |
| B8h | Memory-Mapped I/O Base 7 | | RW | page 60 |
| BCh | Memory-Mapped I/O Limit 7 | | RW | page 61 |
| C0h | PCI I/O Base 0 | | RW | page 62 |
| C4h | PCI I/O Limit 0 | | RW | page 63 |
| C8h | PCI I/O Base 1 | | RW | page 62 |
| CCh | PCI I/O Limit 1 | | RW | page 63 |
| D0h | PCI I/O Base 2 | | RW | page 62 |
| D4h | PCI I/O Limit 2 | | RW | page 63 |
| D8h | PCI I/O Base 3 | | RW | page 62 |
| DCh | PCI I/O Limit 3 | | RW | page 63 |

***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

**Table 3. Function 1 Configuration Registers (Continued)**

| Offset | Register Name | Reset | Access | Description |
|---|---|---|---|---|
| E0h | Configuration Base and Limit 0 | | RW | page 64 |
| E4h | Configuration Base and Limit 1 | | RW | page 64 |
| E8h | Configuration Base and Limit 2 | | RW | page 64 |
| ECh | Configuration Base and Limit 3 | | RW | page 64 |

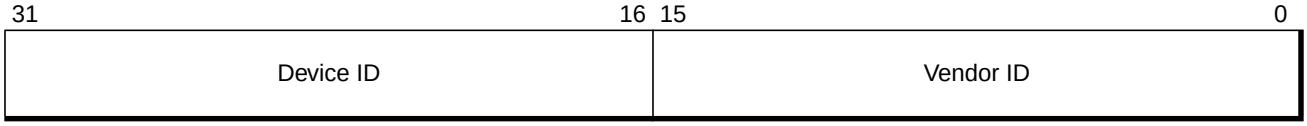
***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

## 3.4.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 1 registers and is part of the standard PCI configuration header.

**Device/Vendor ID Register** **Function 1: Offset 00h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | DevID | Device ID | R | 1101h |
| 15–0 | VenID | Vendor ID | R | 1022h |

### Field Descriptions

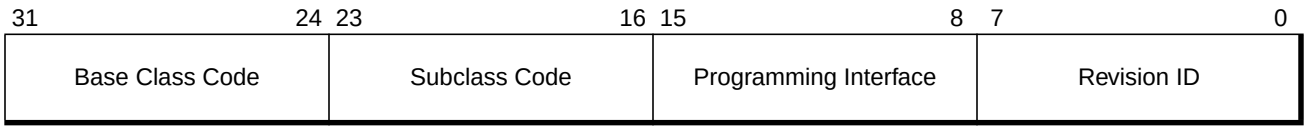
**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1101h for the HyperTransport technology configuration function.

## 3.4.2 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 1 registers and is part of the standard PCI configuration header.

**Class Code/Revision ID Register** **Function 1: Offset 08h**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BCC | Base Class Code | R | 06h |
| 23–16 | SCC | Subclass Code | R | 00h |
| 15–8 | PI | Programming Interface | R | 00h |
| 7–0 | RevID | Revision ID | R | 00h |

### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

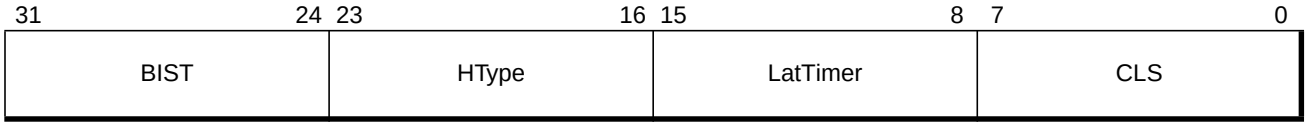
**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

## 3.4.3 Header Type Register

This register specifies the header type for the Function 1 registers and is part of the standard PCI configuration header.

**Header Type Register** **Function 1: Offset 0Ch**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BIST | BIST | R | 00h |
| 23–16 | HType | Header Type | R | 80h |
| 15–8 | LatTimer | Latency Timer | R | 00h |
| 7–0 | CLS | Cache Line Size | R | 00h |

### Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (HType)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

## 3.4.4 DRAM Address Map

These registers define sections of the memory address map for which accesses should be routed to DRAM. DRAM regions must not overlap each other. For addresses within the specified range of a base/limit pair, requests are routed to the memory controller on the node specified by the destination Node ID.

System addresses are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit. For the purposes of this comparison, the lower unspecified bits of the base are assumed to be 0s and the lower unspecified bits of the limit are assumed to be 1s.

An address that maps to both DRAM and memory-mapped I/O will be routed to MMIO.

Programming of the DRAM address maps must be consistent with the Top Of Memory and Memory Type Range registers (see Chapter 12, "Processor Configuration Registers"). Accesses from the CPU can only access the DRAM address maps if the corresponding CPU memory type is DRAM. For accesses from I/O devices, the lookup is based on address only.

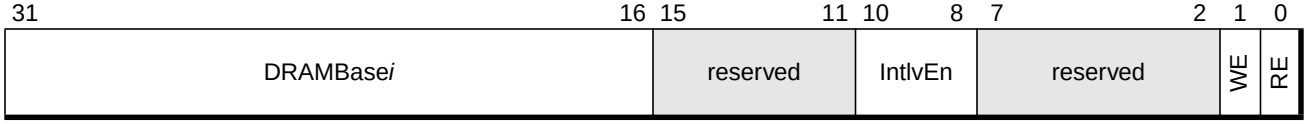
Each base/limit set of DRAM address maps is associated with a particular Node ID (see "Node ID Register" on page 34). The DRAM Base/Limit 0 registers specify the DRAM attached to node 0. Similarly, DRAM on nodes 1–7 is described by base/limit registers 1–7. Note that the destination NodeId field must still be written to ensure correct operation.

When node interleaving is enabled, each node's DRAM limit must be set to the Top Of Memory and each node's DRAM base must be set to 0. The node to which an address is routed to when nodes are interleaved is defined by IntlvEn (Function 1, Offset 40h, 48h, etc.) and IntlvSel (Function 1, Offset 44h, 4Ch, etc.).

Address routed to the DRAM controller (InputAddr) is calculated from the system address (SysAddr) in the following way:
DramAddr[39:0] = {SysAddr[39:24] - DRAMBase[39:24], SysAddr[23:0]},
InputAddr[35:0] = {DramAddr[35:12], DramAddr[11:0]} when node memory is not interleaved,
InputAddr[35:0] = {DramAddr[36:13], DramAddr[11:0]} when 2 nodes are interleaved,
InputAddr[35:0] = {DramAddr[37:14], DramAddr[11:0]} when 4 nodes are interleaved,
InputAddr[35:0] = {DramAddr[38:15], DramAddr[11:0]} when 8 nodes are interleaved.

### 3.4.4.1 DRAM Base *i* Registers

**DRAM Base 0–7 Registers Function 1: Offset 40h, 48h, 50h, 58h, 60h, 68h, 70h, 78h**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | DRAMBase*i* | DRAM Base Address *i* Bits 39–24 | R/W | X |
| 15–11 | reserved | | R | 0 |
| 10–8 | IntlvEn | Interleave Enable | R/W | X |
| 7–2 | reserved | | R | 0 |
| 1 | WE | Write Enable | R/W | 0 |
| 0 | RE | Read Enable | R/W | 0 |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled
1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled
1 = Enabled

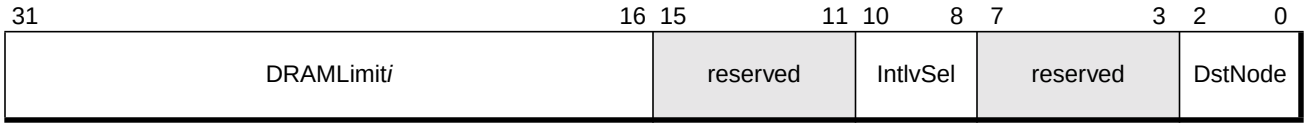
**Interleave Enable (IntlvEn)**—Bits 10–8. This field enables interleaving on a 4-Kbyte boundary between memory on different nodes. The bits are encoded as follows:

000b = No interleave
001b = Interleave on A[12] (2 nodes)
010b = reserved
011b = Interleave on A[12] and A[13] (4 nodes)
100b = reserved
101b = reserved
110b = reserved
111b = Interleave on A[12] and A[13] and A[14] (8 nodes)

**DRAM Base Address *i* Bits 39–24 (DRAMBase*i*)**—Bits 31–16. This field defines the upper address bits of a 40-bit address that defines the start of DRAM region *i* (where *i* = 0, 1, . . . 7).

## 3.4.4.2 DRAM Limit *i* Registers

### DRAM Limit 0–7 Registers Function 1: Offset 44h, 4Ch, 54h, 5Ch, 64h, 6Ch, 74h, 7Ch




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | DRAMLimit*i* | DRAM Limit Address *i* (39–24) | R/W | X |
| 15–11 | reserved | | R | 0 |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 10–8 | IntlvSel | Interleave Select | R/W | X |
| 7–3 | reserved | | R | 0 |
| 2–0 | DstNode | Destination Node ID | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Destination Node ID (DstNode)**—Bits 2–0. This field specifies the node that a packet is sent to if it is within the address range. This field must correspond with the number of the register pair. The bits are encoded as follows:

000b = Node 0 (register set 0)
001b = Node 1 (register set 1)
...
111b = Node 7 (register set 7)

**Interleave Select (IntlvSel)**—Bits 10–8. This field specifies the values of address bits A[14:12] to use with the Interleave Enable field (IntlvEn[2:0]) to determine which 4-Kbyte blocks are routed to this region (See Figure 1).

IntlvSel[0] corresponds to A[12]
IntlvSel[1] corresponds to A[13]
IntlvSel[2] corresponds to A[14]

**DRAM Limit Address *i* (39–24) (DRAMLimit*i*)**—Bits 31–16. This field defines the upper address bits of a 40-bit address that defines the end of DRAM region n (where *i* = 0,1, . . . 7).

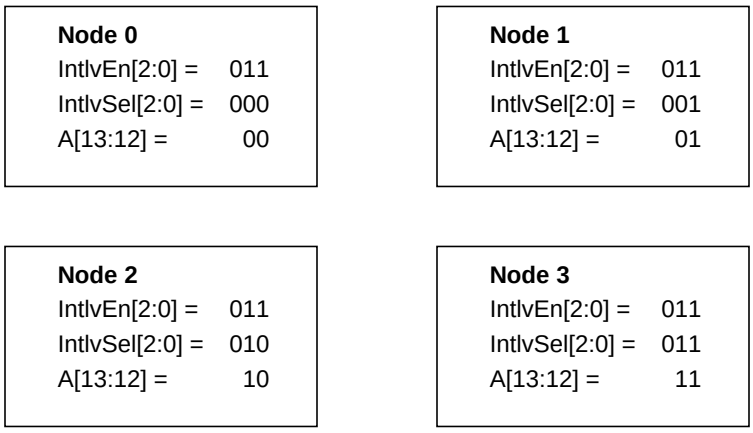



**Figure 1. Interleave Example (IntlvEn Relation to IntlvSel)**

## 3.4.5 Memory-Mapped I/O Address Map Registers

These registers define sections of the memory address map for which accesses should be routed to memory-mapped I/O. MMIO regions must not overlap each other. For addresses within the specified

range of a base/limit pair, requests are routed to the noncoherent HyperTransport link specified by the destination Node ID and destination Link ID.

Addresses are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit. For the purposes of this comparison, the lower unspecified bits of the base are assumed to be 0s and the lower unspecified bits of the limit are assumed to be 1s.

An address that maps to both DRAM and memory-mapped I/O is routed to MMIO.

Programming of the MMIO address maps must be consistent with the Top Of Memory and Memory Type Range registers (see Chapter 12, "Processor Configuration Registers"). In particular, accesses from the CPU can only hit in the MMIO address maps if the corresponding CPU memory type is of type IO. For accesses from I/O devices, the lookup is based on address only.

### 3.4.5.1 Extended Configuration Space Access

Extended configuration space is accessed through MMIO. Instructions used to read extended configuration space are required to take the following form:

```
mov EAX/AX/AL, <any_address_mode>;
```

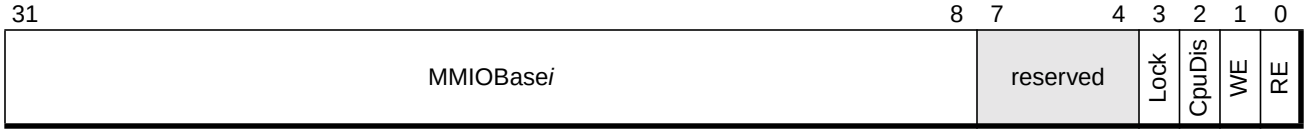
Instructions used to write extended configuration space are required to take the following form:

```
mov <any_address_mode>, EAX/AX/AL;
```

In addition, all such accesses are required not to cross any naturally aligned doubleword boundary. Access to extended configuration registers that do not meet these requirements result in undefined behavior.

### 3.4.5.2 Memory-Mapped I/O Base *i* Registers

**Memory-Mapped I/O Base 0–7 Registers** **Function 1: Offset 80h, 88h, 90h, 98h, A0h, A8h, B0h, B8h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | MMIOBase*i* | Memory-Mapped I/O Base Address *i* (39–16) | R/W | X |
| 7–4 | reserved | | R | 0 |
| 3 | Lock | Lock | R/W | X |
| 2 | CpuDis | CPU Disable | R/W | X |
| 1 | WE | Write Enable | R/W | 0 |
| 0 | RE | Read Enable | R/W | 0 |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled
1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled
1 = Enabled

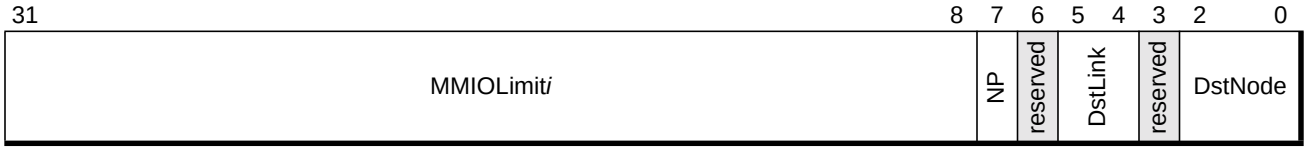
**CPU Disable (CpuDis)**—Bit 2. When set, this bit causes this MMIO range to apply to I/O requests only, not to CPU requests.

**Lock (Lock)**—Bit 3. Setting this bit, along with either the WE or RE bits, makes the base/limit registers read-only.

**Memory-Mapped I/O Base Address *i* (39–16) (MMIOBase*i*)**—Bit 31–8. This field defines the upper address bits of a 40-bit address that defines the start of memory-mapped I/O region n (where n = 0,1, . . . 7).

## 3.4.5.3 Memory-Mapped I/O Limit *i* Registers

**Memory-Mapped I/O Limit *i* Registers** **Function 1: Offset 84h, 8Ch, 94h, 9Ch, A4h, ACh, B4h, BCh**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | MMIOLimit*i* | Memory-Mapped I/O Limit Address *i* | R/W | X |
| 7 | NP | Non-Posted | R/W | X |
| 6 | reserved | | R | 0 |
| 5–4 | DstLink | Destination Link ID | R/W | X |
| 3 | reserved | | R | 0 |
| 2–0 | DstNode | Destination Node ID | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Destination Node ID (DstNode)**—Bits 2–0. This field specifies the node that a packet is sent to if it is within the address range. The bits are encoded as follows:

000b = Node 0
001b = Node 1
...

111b = Node 7

**Destination Link ID (DstLink)**—Bits 5–4. This field specifies the HyperTransport link number to send the packet to once the packet reaches the destination node.

00b = Link 0
01b = Link 1
10b = Link 2
11b = reserved

**Non-Posted (NP)**—Bit 7. When set to 1, this bit forces CPU writes to this memory-mapped I/O region to be non-posted. This may be used to force writes to be non-posted for MMIO regions which map to the legacy ISA/LPC bus, or in conjunction with the DsNpReqLmt field in the HyperTransport Transaction Control register (page 36) in order to allow downstream CPU requests to be counted and thereby limited to a specified number. This latter use of the NP bit may be used to avoid loop deadlock in systems that implement a reflection region in an I/O device that reflects downstream accesses back upstream. See the *HyperTransport™ I/O Link Specification* summary of deadlock scenarios for more information.

0 = CPU writes may be posted
1 = CPU writes must be non-posted

**Memory-Mapped I/O Limit Address *i* (39–16) (MMIOLimit*i*)**—Bits 31–8. This field defines the upper address bits of a 40-bit address that defines the end of memory-mapped I/O region n (where n = 0,1, . . . 7).

## 3.4.6 PCI I/O Address Map Registers

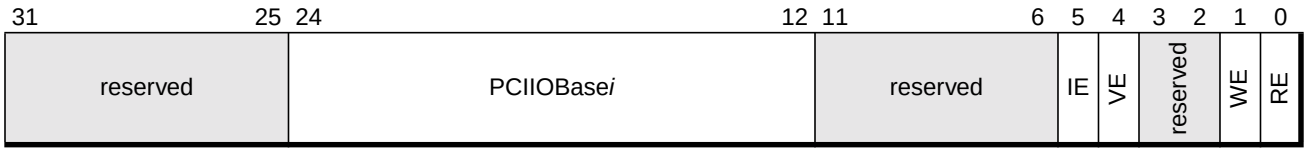
These registers define sections of the PCI I/O address map for which accesses should be routed to each noncoherent HyperTransport technology chain. PCI I/O regions must not overlap each other. For PCI I/O addresses within the specified range of a base/limit pair, requests are routed to the noncoherent HyperTransport link specified by the destination Node ID and destination Link ID.

Addresses are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit. For the purposes of this comparison, the lower unspecified bits of the base are assumed to be 0s and the lower unspecified bits of the limit are assumed to be 1s.

PCI I/O accesses are generated from x86 IN/OUT instructions.

### 3.4.6.1 PCI I/O Base *i* Registers

**PCI I/O Base 0–3 Registers** **Function 1: Offset C0h, C8h, D0h, D8h**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–25 | reserved | | R | 0 |
| 24–12 | PCIIOBase*i* | PCI I/O Base Address *i* | R/W | X |
| 11–6 | reserved | | R | 0 |
| 5 | IE | ISA Enable | R/W | X |
| 4 | VE | VGA Enable | R/W | X |
| 3–2 | reserved | | R | 0 |
| 1 | WE | Write Enable | R/W | 0 |
| 0 | RE | Read Enable | R/W | 0 |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled
1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled
1 = Enabled

**VGA Enable (VE)**—Bit 4. Forces addresses in the first 64 Kbytes of PCI I/O space and where A[9:0] is in the range 3B0–3BBh or 3C0–3DFh to match against this base/limit pair independent of the base/limit addresses (see the *PCI-to-PCI Bridge Architecture* specification for a description of this function). A WE or RE bit must also be set to enable this feature.

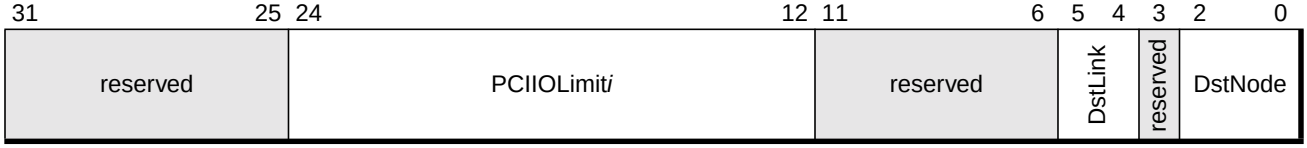
The MMIO 000A_0000h to 000B_FFFFh matching function normally associated with the VGA enable bit in PCI is NOT included in the VE bit. To enable this behavior, an MMIO register pair must be used.

**ISA Enable (IE)**—Bit 5. Blocks addresses in the first 64 Kbytes of PCI I/O space and in the last 768 bytes in each 1-Kbyte block from matching against this base/limit pair (see the *PCI-to-PCI Bridge Architecture* specification for a description of this function). A WE or RE bit must also be set to enable this feature.

**PCI I/O Base Address *i* (PCIIOBase*i*)**—Bits 24–12. This field defines the start of PCI I/O region n (where n = 0, 1, 2,3).

### 3.4.6.2 PCI I/O Limit *i* Registers

**PCI I/O Limit 0–3 Registers** **Function 1: Offset C4h, CCh, D4h, DCh**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–25 | reserved | | R | 0 |
| 24–12 | PCIIOLimit*i* | PCI I/O Limit Address *i* | R/W | X |
| 11–6 | reserved | | R | 0 |
| 5–4 | DstLink | Destination Link ID | R/W | X |
| 3 | reserved | | R | 0 |
| 2–0 | DstNode | Destination Node ID | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Destination Node ID (DstNode)**—Bits 2–0. This field specifies the node to which a packet is sent if it is within the address range. The bits are encoded as follows:

000b = Node 0
001b = Node 1
...
111b = Node 7

**Destination Link ID (DstLink)**—Bits 5–4. This field specifies the link to send the packet to once the packet has reached the destination node.

00b = Link 0
01b = Link 1
10b = Link 2
11b = reserved

**PCI I/O Limit Address *i* (PCI I/OLimit*i*)**—Bits 24–12. This field defines the end of PCI I/O region

## 3.4.7 Configuration Map Registers

These registers define sections of the configuration bus number map for which configuration accesses should be routed to each noncoherent HyperTransport technology chain. Configuration regions must not overlap each other. For configuration accesses with bus numbers within the specified range of a base/limit pair, requests are routed to the noncoherent HyperTransport link specified by the destination Node ID and destination Link ID.

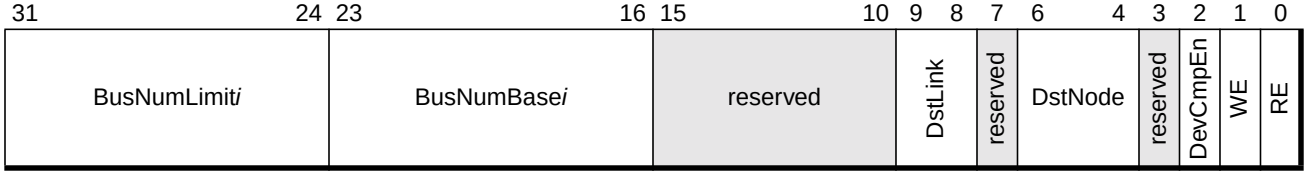
Bus numbers are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit.

In device number compare mode (see DevCmpEn bit), the base/limit fields are interpreted as device number base/limit values for Bus 0 configuration accesses. This may be used to support configurations where Bus 0 is split between two independent noncoherent HyperTransport technology chains.

Configuration accesses to Bus 0, device 24–31, are assumed to be targeting coherent HyperTransport technology devices and are routed to the corresponding coherent HyperTransport node irrespective of

the values in these registers (see Chapter 3, "Memory System Configuration"). Note that the range of device numbers affected may be modified based on the number of nodes present (see "HyperTransport™ Transaction Control Register" on page 36).

### Configuration Base and Limit 0–3 Registers Function 1: Offset E0h, E4h, E8h, ECh




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BusNumLimit*i* | Bus Number Limit *i* | R/W | X |
| 23–16 | BusNumBase*i* | Bus Number Base *i* | R/W | X |
| 15–10 | reserved | | R | 0 |
| 9–8 | DstLink | Destination Link ID | R/W | X |
| 7 | reserved | | R | 0 |
| 6–4 | DstNode | Destination Node ID | R/W | X |
| 3 | reserved | | R | 0 |
| 2 | DevCmpEn | Device Number Compare Enable | R/W | X |
| 1 | WE | Write Enable | R/W | 0 |
| 0 | RE | Read Enable | R/W | 0 |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled
1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled
1 = Enabled

**Device Number Compare Enable (DevCmpEn)**—Bit 2. When this bit is set, this register defines a device number range rather than a bus number range. To match, configuration cycles must be to bus number 0 and have device numbers between BusNumBase and BusNumLimit. This is used to enable multiple noncoherent HyperTransport chains to be configured as Bus 0.

**Destination Node ID (DstNode)**—Bits 6–4. This field specifies the node that a packet is sent to if it is within the address range. The bits are encoded as follows:

000b = Node 0
001b = Node 1
...
111b = Node 7

**Destination Link ID (DstLink)**—Bits 9–8. This field specifies the link to send the packet to once it has reached the destination node and unit.

00b = Link 0
01b = Link 1
10b = Link 2
11b = reserved

**Bus Number Base *i* (BusNumBase*i*)**—Bits 23–16. This field defines the lowest bus number in configuration region *i* (where *i* = 0, 1, 2, 3).

**Bus Number Limit *i* (BusNumLimit*i*)**—Bits 31–24. This bit field defines the highest bus number in configuration region *i* (where *i* = 0, 1, 2, 3).

## 3.5 Function 2—DRAM Controller

Function 2 configuration registers are listed in Table 4.

**Table 4. Function 2 Configuration Registers**

| Offset | Register Name | | | | Reset | Access | Description |
|---|---|---|---|---|---|---|---|
| 00h | Device ID | | Vendor ID (AMD) | | 1102_1022h | RO | page 67 |
| 04h | Status[1] | | Command | | 0000_0000h | RO | |
| 08h | Base Class Code | Subclass Code | Program Interface | Revision ID | 0600_0000h | RO | page 68 |
| 0Ch | BIST | Header Type | Latency Timer | Cache Line Size | 0080_0000h | RO | page 69 |
| 10h | Base Address 0 | | | | 0000_0000h | RO | |
| 14h | Base Address 1 | | | | 0000_0000h | RO | |
| 18h | Base Address 2 | | | | 0000_0000h | RO | |
| 1Ch | Base Address 3 | | | | 0000_0000h | RO | |
| 20h | Base Address 4 | | | | 0000_0000h | RO | |
| 24h | Base Address 5 | | | | 0000_0000h | RO | |
| 28h | Card Bus CIS Pointer | | | | 0000_0000h | RO | |
| 2Ch | Sub-System ID | | Sub-System Vendor ID | | 0000_0000h | RO | |
| 30h | ROM Base Address | | | | 0000_0000h | RO | |
| 34h | Capabilities | | | | 0000_0000h | RO | |
| 38h | reserved | | | | 0000_0000h | RO | |

***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

**Table 4. Function 2 Configuration Registers (Continued)**

| Offset | Register Name | | | | Reset | Access | Description |
|---|---|---|---|---|---|---|---|
| 3Ch | Max Latency | Min GNT | Int Pin | Int Line | 0000_0000h | RO | |
| 40h | DRAM CS Base 0 | | | | 0000_0000h | RW | page 69 |
| 44h | DRAM CS Base 1 | | | | 0000_0000h | RW | page 69 |
| 48h | DRAM CS Base 2 | | | | 0000_0000h | RW | page 69 |
| 4Ch | DRAM CS Base 3 | | | | 0000_0000h | RW | page 69 |
| 50h | DRAM CS Base 4 | | | | 0000_0000h | RW | page 69 |
| 54h | DRAM CS Base 5 | | | | 0000_0000h | RW | page 69 |
| 58h | DRAM CS Base 6 | | | | 0000_0000h | RW | page 69 |
| 5Ch | DRAM CS Base 7 | | | | 0000_0000h | RW | page 69 |
| 60h | DRAM CS Mask 0 | | | | 0000_0000h | RW | page 72 |
| 64h | DRAM CS Mask 1 | | | | 0000_0000h | RW | page 72 |
| 68h | DRAM CS Mask 2 | | | | 0000_0000h | RW | page 72 |
| 6Ch | DRAM CS Mask 3 | | | | 0000_0000h | RW | page 72 |
| 70h | DRAM CS Mask 4 | | | | 0000_0000h | RW | page 72 |
| 74h | DRAM CS Mask 5 | | | | 0000_0000h | RW | page 72 |
| 78h | DRAM CS Mask 6 | | | | 0000_0000h | RW | page 72 |
| 7Ch | DRAM CS Mask 7 | | | | 0000_0000h | RW | page 72 |
| 80h | DRAM Bank Address Mapping | | | | 0000_0000h | RW | page 73 |
| 88h | DRAM Timing Low | | | | 0000_0000h | RW | page 78 |
| 8Ch | DRAM Timing High | | | | 0000_0000h | RW | page 80 |
| 90h | DRAM Configuration Low | | | | 0000_0000h | RW | page 82 |
| 94h | DRAM Configuration High | | | | 0000_0000h | RW | page 85 |
| 98h | DRAM Delay Line | | | | 0000_0000h | RW | page 88 |
| | | | | | | | |

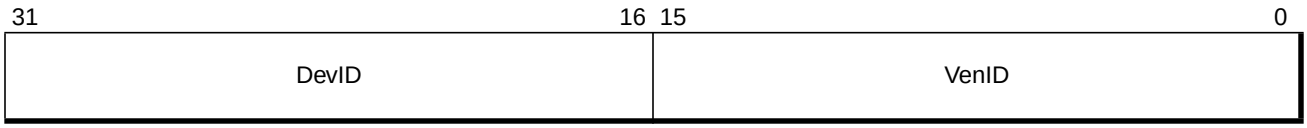
***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

## 3.5.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 2 registers and is part of the standard PCI configuration header.

### Device/Vendor ID Register — Function 2: Offset 00h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | DevID | Device ID | R | 1102h |
| 15–0 | VenID | Vendor ID | R | 1022h |

### Field Descriptions

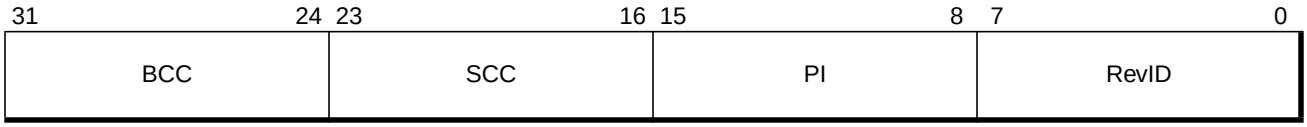
**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1102h for the HyperTransport technology configuration function.

## 3.5.2 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 2 registers and is part of the standard PCI configuration header.

### Class Code/Revision ID Register — Function 2: Offset 08h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BCC | Base Class Code | R | 06h |
| 23–16 | SCC | Subclass Code | R | 00h |
| 15–8 | PI | Programming Interface | R | 00h |
| 7–0 | RevID | Revision ID | R | 00h |

### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

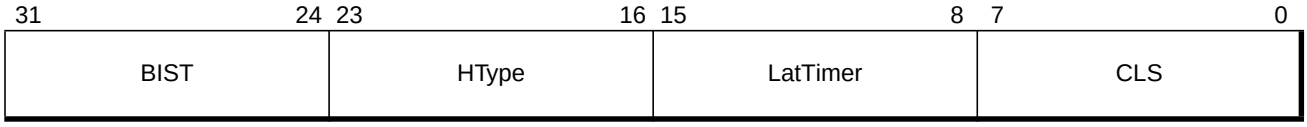
**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

## 3.5.3 Header Type Register

This register specifies the header type for the Function 2 registers and is part of the standard PCI configuration header.

**Header Type Register** **Function 2: Offset 0Ch**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BIST | BIST | R | 00h |
| 23–16 | HType | Header Type | R | 80h |
| 15–8 | LatTimer | Latency Timer | R | 00h |
| 7–0 | CLS | Cache Line Size | R | 00h |

### Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (Htype)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

## 3.5.4 DRAM CS Base Address Registers

These registers define DRAM chip select (CS) address mapping. They are programmed based on data in the Serial Presence Detect (SPD) ROM on each DIMM.

Function 1 address map registers define to which node to route a DRAM request. Function 2 address map registers define what DRAM chip select to access on that particular node. See "DRAM Address Map" on page 57. for more information on addresses routed to the DRAM controller (InputAddr).

Memory size of each chip select bank is defined by a DRAM CS Base Address Register and a DRAM CS Mask register. The chip selects are formed as follows, using the field names from the DRAM CS Base Address Registers and DRAM CS Mask Registers, where

"," means "concatenate"
"/" means "not"

"==" means "equals"
"&" means "and"

ChipSelect[i] is asserted if the following is true:

```
CSBE[i] &
( { InputAddr[35:34], (InputAddr[33:25] & /AddrMaskHi[i][33:25]),
    ( InputAddr[19:13] & /AddrMaskLo[i][19:13]) } ==
  { BaseAddrHi[i][35:34], (BaseAddrHi[i][33:25] & /AddrMaskHi[i][33:25]),
    ( BaseAddrLo[i][19:13] & /AddrMaskLo[i][19:13]) } ) );
```

There are eight DRAM CS Base Address and DRAM CS Mask register pairs. DRAM CS Base Address 0 and DRAM CS Mask 0 define the base address for chip select 0, and so on. DIMM0 is associated with CS0 and CS1, DIMM1 is associated with CS2 and CS3, and so on. If only DIMM2 and DIMM3 are occupied and DIMM2 is connected to CS[5:4] and DIMM3 is connected to CS[7:6], then bit 0 of DRAM CS Base Address Register[7-4] is set to 1 to enable these chip-select banks.

**Table 5. DRAM CS Base Address and DRAM CS Mask Registers**

| Registers | Chip Selects | Memory Module D[127:64][1] | Memory Module D[63:0] |
|---|---|---|---|
| DRAM CS Base and Mask 0 | CS0 | Extended DIMM0 | DIMM0 |
| DRAM CS Base and Mask 1 | CS1 | Extended DIMM0 | DIMM0 |
| DRAM CS Base and Mask 2 | CS2 | Extended DIMM1 | DIMM1 |
| DRAM CS Base and Mask 3 | CS3 | Extended DIMM1 | DIMM1 |
| DRAM CS Base and Mask 4 | CS4 | Extended DIMM2 | DIMM2 |
| DRAM CS Base and Mask 5 | CS5 | Extended DIMM2 | DIMM2 |
| DRAM CS Base and Mask 6 | CS6 | Extended DIMM3 | DIMM3 |
| DRAM CS Base and Mask 7 | CS7 | Extended DIMM3 | DIMM3 |

***Notes:***

1. *This column only applies to the AMD Opteron™ processor configured for a 128-bit DRAM interface.*

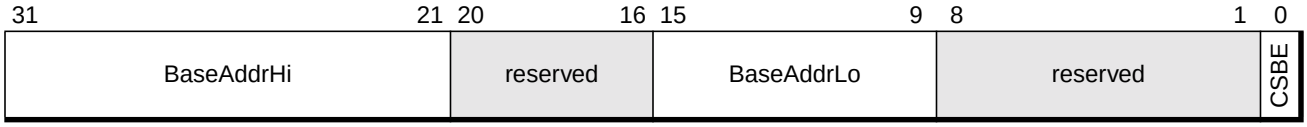
DRAM memory can be assigned to chip select banks in two ways: non-interleaving, when contiguous addresses are assigned to each chip select bank, and interleaving, when non-contiguous addresses are assigned to each chip select bank.

Non-interleaving mode can always be used. The BIOS must assign the largest DIMM chip-select range to the lowest address. As addresses increase, the chip select size must remain constant or decrease. This is necessary to keep DIMM chip select banks on aligned address boundaries as chip-select banks with different depths are added. The masking does not work otherwise.

Memory interleaving mode can be used if the memory system is composed of a single type and size DRAM, and if the number of chip selects is a power of two. DRAM controller swaps low order address bits with high order address bits during decoding. As a result, some low order address bits are used to determine chip select, and some high order address bits are used to determine DIMM row. It allows that a switch between DIMMs occurs before a page conflict within a DIMM. The one cycle turnaround saves time needed to close/open new pages. Algorithm for programming DRAM CS Base

Address and DRAM CS Mask registers in the interleaving mode is described in section "DRAM Address Mapping in Interleaving Mode" on page 76.

### DRAM CS Base Address Registers — Function 2: Offset 40h, 44h, 48h, 4Ch, 50h, 54h, 58h, 5Ch




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–21 | BaseAddrHi | Base Address (35–25) | R/W | 0 |
| 20–16 | reserved | | R | 0 |
| 15–9 | BaseAddrLo | Base Address (19–13) | R/W | 0 |
| 8–1 | reserved | | R | 0 |
| 0 | CSBE | Chip-Select Bank Enable | R/W | 0 |

### Field Descriptions

**Chip-Select Bank Enable (CSBE)**—Bit 0. This bit enables access to the defined address space.

**Base Address (19–13) (BaseAddrLo)**—Bits 15–9. This field is only used for memory interleaving to avoid page conflicts. A new chip select bank is accessed before accessing a new row in the old chip select bank, delaying or avoiding a page conflict.

If contiguous addresses from address 0 are accessed, the controller would first access a row (e.g., row 0) in bank 0, chip select 0. As the address increases, the controller would access different columns in the following order:

row 0/bank 0/chip select 0 and then
row 0/bank 1/chip select 0 and then
row 0/bank 2/chip select 0 and then
row 0/bank 3/chip select 0 and then
row 0/bank 0/chip select 1.

A chip select bank does not have a contiguous region of memory assigned to it. Memory is interleaved between two DIMMs, four DIMMs, or eight DIMMs.

**Base Address (35–25) (BaseAddrHi)**—Bits 31–21. These bits decode 32-Mbyte blocks of memory. In the non-interleaving mode (BaseAddrLo has a value of 0), physical addresses map to DRAM addresses in the following way:

Col—Lowest order physical address bits
Bank—Second lowest order physical address bits (page miss is better than page conflict)
Row—Second highest order physical address bits (page conflict before accessing new chip)
Chip Select—Highest order physical address bits

In the non-interleaving mode, contiguous addresses from 0 would first access a row (e.g., row 0) in bank 0, chip select 0. As the address increases, the next access is to a different column and eventually a different chip select bank, e.g.:

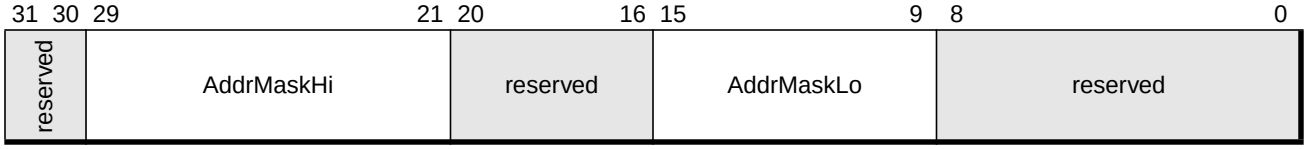
row 0/bank 0/chip select 0 and then
row 0/bank 1/chip select 0 and then
row 0/bank 2/chip select 0 and then
row 0/bank 3/chip select 0 and then
row 1/bank 0/chip select 0

Memory controller accesses all rows and banks in a single chip select bank before accessing a new chip select bank in the non-interleaving mode.

## 3.5.5 DRAM CS Mask Registers

The purpose of this register is to exclude the corresponding address bits from the comparison with the DRAM Base address register.

### DRAM CS Mask Registers Function 2: Offset 60h, 64h, 68h, 6Ch, 70h, 74h, 78h, 7Ch




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–30 | reserved | | R | 0 |
| 29–21 | AddrMaskHi | Address Mask (33–25) | R/W | 0 |
| 20–16 | reserved | | R | 0 |
| 15–9 | AddrMaskLo | Address Mask (19–13) | R/W | 0 |
| 8–0 | reserved | | R | 0 |

### Field Descriptions

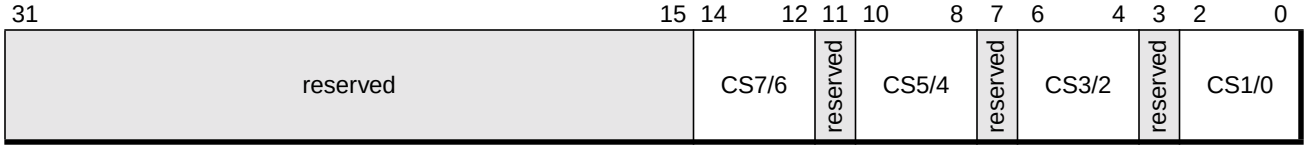
**Address Mask (19–13) (AddrMaskLo)**—Bits 15–9. This field specifies the addresses to be excluded for the memory interleaving mode described in the Base Address bit definitions.

**Address Mask (33–25) (AddrMaskHi)**—Bits 29–21. This field defines the top Address Mask bits. The bits with an address mask of 1 are excluded from the address comparison. This allows the memory block size to be larger than 32 Mbytes. If Address Mask bit 25 is set to 1, the memory block size is 64 Mbytes.

## 3.5.6 DRAM Bank Address Mapping Register

The DIMM may have one or two chip-select banks, but it always has support for up to two chip-select banks. The address mode covers the whole DIMM, regardless of whether there is one or two chip-select banks.

**DRAM CS Address Mapping Register** **Function 2: Offset 80h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–15 | reserved | | R | 0 |
| 14–12 | CS7/6 | CS7/6 | R/W | 0 |
| 11 | reserved | | R | 0 |
| 10–8 | CS5/4 | CS5/4 | R/W | 0 |
| 7 | reserved | | R | 0 |
| 6–4 | CS3/2 | CS3/2 | R/W | 0 |
| 3 | reserved | | R | 0 |
| 2–0 | CS1/0 | CS1/0 | R/W | 0 |

### Field Descriptions

**Chip Select 1/0 (CS1/0)**—Bits 2–0. This field specifies the memory module size. This field is programmed the same regardless of whether the DRAM interface is 64-bits or 128-bits wide. This field describes the type of DIMMs that compose the chip select.

Chip Select Bank Address Mode Encoding:

000b = 32 Mbyte (Rows =12 & Col=8)
001b = 64 Mbyte (Rows =12 & Col=9)
010b = 128 Mbyte (Rows =13 & Col=9) | (Rows=12 & Col=10)
011b = 256 Mbyte (Rows =13 & Col=10) | (Rows=12 & Col=11)
100b = 512 Mbyte (Rows =13 & Col=11) | (Rows=14 & Col=10)
101b = 1 Gbyte (Rows =14 & Col=11) | (Rows=13 & Col=12)
110b = 2 Gbyte (Rows =14 & Col=12)
111b = reserved

**Chip Select 3/2 (CS3/2)**—Bits 6–4. This field specifies the memory module size. The bits are encoded as shown in CS1/0.

**Chip Select 5/4 (CS5/4)**—Bits 10–8. This field specifies the memory module size. The bits are encoded as shown in CS1/0.

**Chip Select 7/6 (CS7/6)**—Bits 14–12. This field specifies the memory module size. The bits are encoded as shown in CS1/0.

### 3.5.6.1 DRAM Address Mapping in Non-Interleaving Mode

The address line mapping for non-interleaving mode with a 64-bit DRAM interface is shown in Table 6 and with a 128-bit DRAM interface in Table 7. Column A10 in both tables presents the precharge all signal (PC).

**Table 6. Mapping Host Address Lines to Memory Address Lines (64-Bit Interface)**

**64-Bit Interface to Memory**

| Reg 80h CSn[2:0] | | B1 | B0 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000b 64Mb x16, 32MB | Row<br>Col | 12<br>12 | 11<br>11 | X<br>X | X<br>X | 18<br>X | 17<br>PC | 16<br>X | 15<br>X | 14<br>10 | 13<br>9 | 24<br>8 | 23<br>7 | 22<br>6 | 21<br>5 | 20<br>4 | 19<br>3 |
| 001b 128Mb x16/ 64Mb x8, 64MB | Row<br>Col | 12<br>12 | 13<br>13 | X<br>X | X<br>X | 18<br>X | 17<br>PC | 16<br>X | 15<br>11 | 14<br>10 | 25<br>9 | 24<br>8 | 23<br>7 | 22<br>6 | 21<br>5 | 20<br>4 | 19<br>3 |
| 010b 64Mb x4/ 256Mb x16/ 128Mb x8, 128MB | Row<br>Col | 12<br>12 | 13<br>13 | X<br>X | 26<br>X | 18<br>X | 17<br>PC | 16<br>26 | 15<br>11 | 14<br>10 | 25<br>9 | 24<br>8 | 23<br>7 | 22<br>6 | 21<br>5 | 20<br>4 | 19<br>3 |
| 011b 128Mb x4/ 512Mb x16/ 256Mb x8, 256MB | Row<br>Col | 14<br>14 | 13<br>13 | X<br>X | 27<br>X | 18<br>27 | 17<br>PC | 16<br>12 | 15<br>11 | 26<br>10 | 25<br>9 | 24<br>8 | 23<br>7 | 22<br>6 | 21<br>5 | 20<br>4 | 19<br>3 |
| 100b 256Mb x4/ 512Mb x8/ 1Gb x16, 512MB | Row<br>Col | 14<br>14 | 13<br>13 | 28<br>X | 27<br>X | 18<br>28 | 17<br>PC | 16<br>12 | 15<br>11 | 26<br>10 | 25<br>9 | 24<br>8 | 23<br>7 | 22<br>6 | 21<br>5 | 20<br>4 | 19<br>3 |
| 101b 512Mb x4/ 1Gb x8, 1024MB | Row<br>Col | 14<br>14 | 15<br>15 | 28<br>X | 27<br>28 | 18<br>13 | 17<br>PC | 16<br>12 | 29<br>11 | 26<br>10 | 25<br>9 | 24<br>8 | 23<br>7 | 22<br>6 | 21<br>5 | 20<br>4 | 19<br>3 |
| 110b 1Gb x4 2048MB | Row<br>Col | 14<br>14 | 15<br>15 | 28<br>X | 27<br>30 | 18<br>13 | 17<br>PC | 16<br>12 | 29<br>11 | 26<br>10 | 25<br>9 | 24<br>8 | 23<br>7 | 22<br>6 | 21<br>5 | 20<br>4 | 19<br>3 |

**Table 7. Mapping Host Address Lines to Memory Address Lines (128-Bit Interface)**

**128-Bit Interface to Memory**

| Reg 80h CSn[2:0] | | B1 | B0 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000b 64Mb x16, 32MB | Row Col | 13 13 | 12 12 | X X | X X | 19 X | 18 PC | 17 X | 16 X | 15 11 | 14 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 |
| 001b 128Mb x16/ 64Mb x8, 64MB | Row Col | 13 13 | 14 14 | X X | X X | 19 X | 18 PC | 17 X | 16 12 | 15 11 | 26 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 |
| 010b 64Mb x4/ 256Mb x16/ 128Mb x8, 128MB | Row Col | 13 13 | 14 14 | X X | 27 X | 19 X | 18 PC | 17 27 | 16 12 | 15 11 | 26 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 |
| 011b 128Mb x4/ 512Mb x16/ 256Mb x8, 256MB | Row Col | 15 15 | 14 14 | X X | 28 X | 19 28 | 18 PC | 17 13 | 16 12 | 27 11 | 26 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 |
| 100b 256Mb x4/ 512Mb x8/ 1Gb x16, 512MB | Row Col | 15 15 | 14 14 | 29 X | 28 X | 19 29 | 18 PC | 17 13 | 16 12 | 27 11 | 26 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 |
| 101b 512Mb x4/ 1Gb x8, 1024MB | Row Col | 15 15 | 16 16 | 29 X | 28 29 | 19 14 | 18 PC | 17 13 | 30 12 | 27 11 | 26 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 |
| 110b 1Gb x4 2048MB | Row Col | 15 15 | 16 16 | 29 X | 28 31 | 19 14 | 18 PC | 17 13 | 30 12 | 27 11 | 26 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 |

The BIOS sets up the base address and base mask registers after determining the number of populated DIMMs, how many chip-select banks they have, and their sizes. A single base address and mask are initialized for each chip select. Hardware decodes addresses and determines to which DIMM chip-select range the address maps, as a function of the base address and mask.

**Example.** DRAM memory consists of:

- DIMM0, 2-sided with 128 MBytes on each side (total DIMM0 memory is 256 Mbytes)
- DIMM1, 2-sided with 256 MBytes on each side (total DIMM1 memory: 512 Mbyte)
- DIMM2, 2-sided with 64 MBytes on each side (total DIMM2 memory: 128 Mbyte)
- DIMM3, 2-sided with 128 MBytes on each side (total DIMM3 memory: 256 Mbyte)

A 64-bit interface to DRAM is used.

Because two of the DIMMs are the same size, there are two different ways to program the base address and base mask registers. One way to program them is as follows:

Function 2, Offset 80h = 0000_2132h // CS0/1 = 128 MB; CS2/3 = 256 MB; CS4/5 = 64 MB; CS6/7 = 128 MB

Function 2, Offset 5Ch = 0380_0001h // 896 MB base = 768 MB + 128 MB

Function 2, Offset 58h = 0300_0001h // 768 MB base = 640 MB + 128 MB

Function 2, Offset 54h = 0440_0001h // 1088 MB base = 1024 MB + 64 MB

Function 2, Offset 50h = 0400_0001h // 1024 MB base = 896 MB + 128 MB

Function 2, Offset 4Ch = 0100_0001h // 256 MB base = 0 MB + 256 MB

Function 2, Offset 48h = 0000_0001h // 0 MB base

Function 2, Offset 44h = 0280_0001h // 640 MB base = 512 MB + 128 MB

Function 2, Offset 40h = 0200_0001h // 512 MB base = 256 MB + 256 MB

Function 2, Offset 7Ch = 0060_FE00h // CS7 = 128 MB

Function 2, Offset 78h = 0060_FE00h // CS6 = 128 MB

Function 2, Offset 74h = 0020_FE00h // CS5 = 64 MB

Function 2, Offset 70h = 0020_FE00h // CS4 = 64 MB

Function 2, Offset 6Ch = 00E0_FE00h // CS3 = 256 MB

Function 2, Offset 68h = 00E0_FE00h // CS2 = 256 MB

Function 2, Offset 64h = 0060_FE00h // CS1 = 128 MB

Function 2, Offset 60h = 0060_FE00h // CS0 = 128 MB

### 3.5.6.2 DRAM Address Mapping in Interleaving Mode

In memory interleaving mode all DIMM chip-select ranges are the same size and type, and the number of chip selects is a power of two. A BIOS algorithm for programming DRAM CS Base Address and DRAM CS Mask registers in memory interleaving mode is:

1. Program all DRAM CS Base Address and DRAM CS Mask registers using contiguous mapping.
2. For each enabled chip select, swap BaseAddrHi bits with BaseAddrLo bits, defined in Table 8 and Table 9.

3. For each enabled chip select, swap AddrMaskHi bits with AddrMaskLo bits, defined in Table 8 and Table 9.

**Table 8. Swapped physical address lines for interleaving with 64-bit interface**

| Chip Select Size | Swapped Base Address and Address Mask bits | | |
|---|---|---|---|
| | 8 way interleaving | 4 way interleaving | 2 way interleaving |
| 32-MByte | [27:25] and [15:13] | [26:25] and [14:13] | [25] and [13] |
| 64-MByte | [28:26] and [16:14] | [27:26] and [15:14] | [26] and [14] |
| 128-MByte | [29:27] and [16:14] | [28:27] and [15:14] | [27] and [14] |
| 256-MByte | [30:28] and [17:15] | [29:28] and [16:15] | [28] and [15] |
| 512-MByte | [31:29] and [17:15] | [30:29] and [16:15] | [29] and [15] |
| 1-GByte | [32:30] and [18:16] | [31:30] and [17:16] | [30] and [16] |
| 2-GByte | [33:31] and [18:16] | [32:31] and [17:16] | [31] and [16] |

**Table 9. Swapped physical address lines for interleaving with 128-bit interface**

| Chip Select Size | Swapped Base Address and Address Mask bits | | |
|---|---|---|---|
| | 8 way interleaving | 4 way interleaving | 2 way interleaving |
| 64-MByte | [28:26] and [16:14] | [27:26] and [15:14] | [26] and [14] |
| 128-MByte | [29:27] and [17:15] | [28:27] and [16:15] | [27] and [15] |
| 256-MByte | [30:28] and [17:15] | [29:28] and [16:15] | [28] and [15] |
| 512-MByte | [31:29] and [18:16] | [30:29] and [17:16] | [29] and [16] |
| 1-GByte | [32:30] and [18:16] | [31:30] and [17:16] | [30] and [16] |
| 2-GByte | [33:31] and [19:17] | [32:31] and [18:17] | [31] and [17] |
| 4-GByte | Not implemented. | [33:32] and [18:17] | [32] and [17] |

**Example.** DRAM memory consists of two 2-sided DIMMs with 256 MBytes on each side. A 64-bit interface to DRAM is used.

1. Register settings for contiguous memory mapping are:

   Function 2, Offset 80h = 0000_0033h // CS0/1 = 256 MB; CS2/3 = 256 MB

   Function 2, Offset 40h = 0000_0001h // 0 MB base

   Function 2, Offset 44h = 0100_0001h // 256 MB base = 0 MB + 256 MB

   Function 2, Offset 48h = 0200_0001h // 512 MB base = 256 MB + 256 MB

   Function 2, Offset 4Ch = 0300_0001h // 768 MB base = 512 MB + 256 MB

   Function 2, Offset 60h = 00e0_fe00h // CS0 = 256 MB

   Function 2, Offset 64h = 00e0_fe00h // CS1 = 256 MB

Function 2, Offset 68h = 00e0_fe00h // CS2 = 256 MB

Function 2, Offset 6Ch = 00e0_fe00h // CS3 = 256 MB

2. Base Address bits to be swapped are defined inTable 8 (64-bit interface), 256MByte row (chip select size), 4 way interleaving column (4 chip selects are used). Base Address bits [29:28] are defined with BaseAddrHi[25:24]. Base Address bits [16:15] are defined with BaseAddrLo[12:11].

   Function 2, Offset 40h=0000_0001h

   Function 2, Offset 44h=0000_0801h

   Function 2, Offset 48h=0000_1001h

   Function 2, Offset 4Ch=0000_1801h

3. Address Mask bits to be swapped are the same as the Base Address bits defined in the previous step. Address Mask bits [29:28] are defined with AddrMaskHi[25:24]. Address Mask bits [16:15] are defined with AddrMaskLo[12:11].

   Function 2, Offset 60h=03e0_e600h

   Function 2, Offset 64h=03e0_e600h

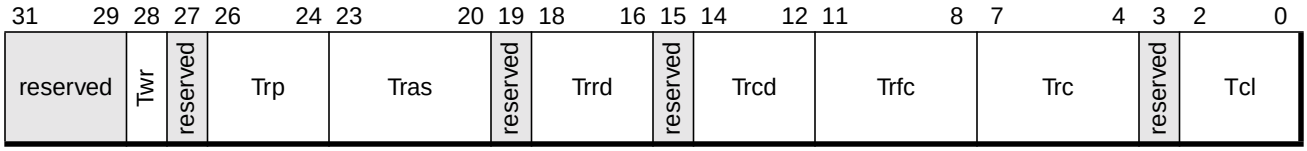
   Function 2, Offset 68h=03e0_e600h

   Function 2, Offset 6Ch=03e0_e600h

## 3.5.7 DRAM Timing Low Register

This register contains timing parameters specified in a DRAM data sheet.

**DRAM Timing Low Register** **Function 2: Offset 88h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–29 | reserved | | R | 0 |
| 28 | Twr | Write Recovery Time | R/W | 0 |
| 27 | reserved | | R | 0 |
| 26–24 | Trp | Row Precharge Time | R/W | 0 |
| 23–20 | Tras | Minimum RAS# Active Time | R/W | 0 |
| 19 | reserved | | R | 0 |
| 18–16 | Trrd | Active-to-active (RAS#-to-RAS#) Delay | R/W | 0 |
| 15 | reserved | | R | 0 |
| 14–12 | Trcd | RAS#-active to CAS#-read/write Delay | R/W | 0 |
| 11–8 | Trfc | Row Refresh Cycle Time | R/W | 0 |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 7–4 | Trc | Row Cycle Time | R/W | 0 |
| 3 | reserved | | R | 0 |
| 2–0 | Tcl | CAS# Latency | R/W | 0 |

## Field Descriptions

**CAS# Latency (Tcl)**—Bits 2–0. Specifies the CAS#-to-read-data-valid.

000b = reserved
001b = CL=2
010b = CL=3
011b = reserved
100b = reserved
101b = CL=2.5
110b = reserved
111b = reserved

**Row Cycle Time (Trc)**—Bits 7–4. RAS#-active to RAS#-active or auto refresh of the same bank. Typically ~70 ns. These bits are encoded as follows (only 7–13 are valid; all others are reserved):

0000b = 7 bus clocks
0001b = 8 bus clocks
. . .
1110b = 21 bus clocks
1111b = 22 bus clocks

**Row Refresh Cycle Time (Trfc)**—Bits 11–8. Auto-refresh-active to RAS#-active or RAS# auto-refresh.

0000b = 9 bus clocks
0001b = 10 bus clocks
...
1110b = 23 bus clocks
1111b = 24 bus clocks

**RAS#-active to CAS#-read/write Delay (Trcd)**—Bits 14–12. Specifies the RAS#-active to CAS#-read/write delay to the same bank.

000b = reserved
001b = reserved
010b = 2 bus clocks
011b = 3 bus clocks
100b = 4 bus clocks
101b = 5 bus clocks
110b = 6 bus clocks
111b = reserved

**Active-to-active (RAS#-to-RAS#) Delay (Trrd)**—Bits 18–16. Specifies the active-to-active delay (RAS#-to-RAS#) of different banks.

000b = reserved
001b = reserved
010b = 2 bus clocks
011b = 3 bus clocks
100b = 4 bus clocks
101b = reserved
110b = reserved
111b = reserved

**Minimum RAS# Active Time (Tras)**—Bits 23–20. Specifies the minimum RAS# active time.

0000b to 0100b = reserved
0101b = 5 bus clocks
...
1111b = 15 bus clocks

**Row Precharge Time (Trp)**—Bits 26–24. Specifies the Row precharge Time. (Precharge-to-Active or Auto-Refresh of the same bank.

000b = reserved
001b = reserved
010b = 2 bus clocks
011b = 3 bus clocks
100b = 4 bus clocks
101b = 5 bus clocks
110b = 6 bus clocks
111b = reserved

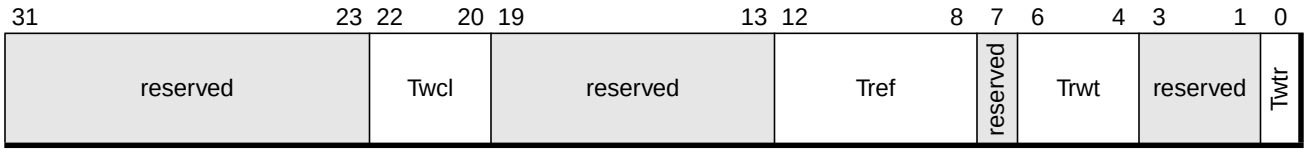
**Write Recovery Time (Twr)**—Bit 28. Measures when the last write datum is safely registered by the DRAM. It measures from the last data to precharge (writes can go back-to-back).

0 = 2 bus clocks
1 = 3 bus clocks

## 3.5.8 DRAM Timing High Register

This register contains the normal timing parameters specified in a DRAM data sheet.

**DRAM Timing High Register** **Function 2: Offset 8Ch**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–23 | reserved | | R | 0 |
| 22–20 | Twcl | Write CAS Latency | R/W | 0 |
| 19–13 | reserved | | R | 0 |
| 12–8 | Tref | Refresh Rate | R/W | 0 |
| 7 | reserved | | R | 0 |
| 6–4 | Trwt | Read-to-Write Delay | R/W | 0 |
| 3–1 | reserved | | R | 0 |
| 0 | Twtr | Write-to-Read Delay | R/W | 0 |

## Field Descriptions

**Write-to-Read Delay (Twtr)**—Bit 0. This bit specifies the write-to-read delay. It is measured from the rising edge following the last non-masked data strobe to the rising edge of the next Read command.

0 = 1 bus Clocks
1 = 2 bus Clocks

**Read-to-Write Delay (Trwt)**—Bits 6–4. Specifies the read-to-write delay.

This is not a DRAM-specified timing parameter, but must be considered due to routing latencies on the clock forwarded bus. It is counted from the first address bus slot that was not associated with part of the read burst. These bits are encoded as follows:

000b = 1 bus clocks
001b = 2 bus clocks
010b = 3 bus clocks
011b = 4 bus clocks
100b = 5 bus clocks
101b = 6 bus clocks
110b = reserved
111b = reserved

**Refresh Rate (Tref)**—Bits 12–8. Specifies the refresh rate of the DIMM requiring the most frequent refresh. These bits are encoded as follows:

00000b = 100 MHz 4K rows
00001b = 133 MHz 4K rows
00010b = 166 MHz 4K rows
00011b = 200 MHz 4K rows
01000b = 100 MHz 8K/16K rows
01001b = 133 MHz 8K/16K rows
01010b = 166 MHz 8K/16K rows
01011b = 200 MHz 8K/16K rows

**Write CAS Latency (Twcl)**—Bits 22–20. Specifies the write CAS latency. Unbuffered DIMMs should be programmed as 0 and registered DIMMs as 1. These bits are encoded as follows:

000b = 1 Mem clock after CAS#
001b = 2 Mem clocks after CAS#

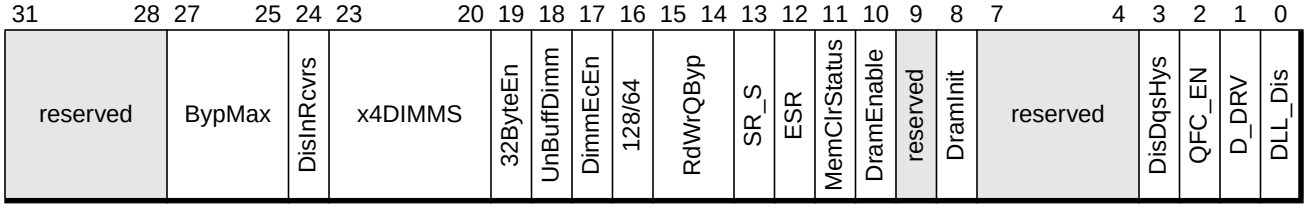
## 3.5.9 DRAM Configuration Registers

The following registers contain all the bits to configure the DRAMs for proper operation.

### 3.5.9.1 DRAM Configuration Low Register

This register controls drive strength, refresh, and initialization.

**DRAM Configuration Low Register** **Function 2: Offset 90h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–28 | reserved | | R | 0 |
| 27–25 | BypMax | Bypass Max | R/W | 0 |
| 24 | DisInRcvrs | Disable DRAM Receivers | R/W | 0 |
| 23–20 | x4DIMMS | x4DIMMS | R/W | 0 |
| 19 | 32ByteEn | Enable 32-Byte Granularity | R/W | 0 |
| 18 | UnBuffDimm | Unbuffered DIMMs | R/W | 0 |
| 17 | DimmEcEn | DIMM ECC Enable | R/W | 0 |
| 16 | 128/64 | 128-Bit/64-Bit | R/W | 0 |
| 15–14 | RdWrQByp | Read/Write Queue Bypass Count | R/W | 0 |
| 13 | SR_S | Self-Refresh Status | R/W | 0 |
| 12 | ESR | Exit Self-Refresh | R/W | 0 |
| 11 | MemClrStatus | Memory Clear Status | R | 0 |
| 10 | DramEnable | DRAM Enable | R | 0 |
| 9 | reserved | SBZ | R | 0 |
| 8 | DramInit | DramInit | R/W | 0 |
| 7–4 | reserved | | R | 0 |
| 3 | DisDqsHys | Disable DQS Hysteresis | R/W | 0 |
| 2 | QFC_EN | QFC_EN Enable | R/W | 0 |
| 1 | D_DRV | D_DRV | R/W | 0 |
| 0 | DLL_Dis | DLL Disable | R/W | 0 |

### Field Descriptions

**DLL Disable (DLL_Dis)**—Bit 0. Disables the Digital Locked Loop that controls relationship between the memory Data and the memory strobes (DQS).

0 = Enabled (default)
1 = Disabled

**D_DRV (D_DRV)**—Bit 1. Controls the drive strength of the DRAM control lines. This bit is required by the DRAM in the Extended Memory Register Set (EMRS*)* command. It applies to all outputs.

0 = Normal Drive (default)
1 = Weak Drive (beware)

**QFC_EN Enable (QFC_EN)**—Bit 2. Causes the corresponding enable bit to be set in the (external) DRAM extended mode register. The QFC signal is an optional DRAM output control used to isolate module loads (DIMMs) from the system memory bus by means of FET switches when the given module is not being accessed.

0 = Disabled (default)
1 = Enabled

**Disable DQS Hysteresis (DisDqsHys)**—Bit 3. Disables DQS input filtering when set to 1. It is required that BIOS (1) set this bit high prior to initializing DRAM (through bit 8, DramInit) and (2), in a subsequent access to this register, clear this bit; the write that clears this bit may be concurrent with the write that sets DramInit or it may be prior to the write that sets DramInit.

**DramInit (DramInit)**—Bit 8. Set by the BIOS to initiate the DRAM initialization sequence. It is cleared by the DRAM controller when the initialization completes. Bits 3–0 and Tcl must be properly programmed before (or at same time as) the bit is set to 1. See bit 3 of this register, DisDqsHys, for BIOS programming requirements related to DramInit.

0 = Initialization done or not started (default)

**DRAM Enable (DramEn)**—Bit 10. When set, this bit indicates that DRAM is enabled. This bit is set by hardware at the completion of DRAM initialization or on an exit from self refresh. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**Memory Clear Status (MemClrStatus)**—Bit 11. When set, this bit indicates that the memory clear function is complete. It is only cleared by reset. BIOS should not write or read the DRAM until this bit is set by hardware. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**Exit Self-Refresh (ESR)**—Bit 12. BIOS sets either the DramInit bit (bit 8) or the Exit Self-Refresh bit (bit 12) after reset, depending on whether the part is coming out of Suspend-to-RAM mode or not (bit 12 is set when coming out of suspend-to-RAM mode). After this bit is set, it reads back as a 1 until the process of exiting self refresh is complete, at which point it reads back as a 0.

**Self-Refresh Status (SR_S)**—Bit 13. When asserted, indicates that the DRAMs are in self-refresh mode. When it deasserts, indicates that the memory controller is ready to process requests again. This bit may be polled until the DRAMs exit the self-refresh state. This bit cannot accurately reflect that the DRAM is in self-refresh when resuming from suspend-to-RAM (STR) because the processor loses power in the STR state. Therefore, this bit must be set, along with bit 12, to properly exit suspend-to-RAM mode.

0 = Normal operation (default)
1 = Self-refresh mode active

**Read/Write Queue Bypass Count (RdWrQByp)**—Bits 15–14. Specifies the number of times the oldest operation in the DCI read/write queue can be bypassed before the arbiter is overridden and the oldest operation is chosen.

00b = 2
01b = 4
10b = 8
11b = 16

**128-Bit/64-Bit (128/64)**—Bit 16. Indicates a 128-bit interface to DRAM.

0 = 64-bit interface to DRAM
1 = 128-bit interface to DRAM

**DIMM ECC Enable (DimmEcEn)**—Bit 17. When DimmEcEn is set to 1, it indicates that all enabled DIMMs have support for ECC check bit storage.

0 = Some DIMMs do not have ECC bits
1 = All DIMMs have ECC bits

**Unbuffered DIMMs (UnBuffDimm)**—Bit 18. When asserted, the DRAM controller is only connected to unbuffered DIMMs. In this case the controller drives address/control 3/4 MEMCLK earlier than the clock to account for the address flight time. If deasserted, the controller drives address/control 1/2 MEMCLK earlier than clock.

0 = Buffered DIMMs
1 = Unbuffered DIMMs

**Enable 32-Byte Granularity (32ByteEn)**—Bit 19. When asserted, indicates that the burst counter should be chosen to optimize data bus bandwidth for 32-byte accesses (except when the bus width is 128 bits. This bit, along with bit 16, produces the following truth table.

| Bit 19 | Bit 16 | |
|---|---|---|
| 0 | 0 | 8-beat bursts (64 bytes) |
| 0 | 1 | 4-beat bursts (64 bytes) |
| 1 | 0 | 4-beat bursts (32 bytes) |
| 1 | 1 | 4-beat bursts (64 bytes) |

**x4DIMMS[3:0]**—Bits 23–20. Indicates whether each of the four DIMMs are x4 or not.

0 = DIMM is not x4
1 = x4 DIMM present

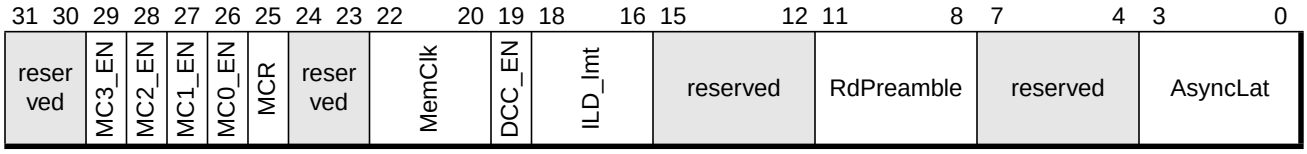
**Disable DRAM Receivers (DisInRcvrs)**—Bit 24. When the memory controller is disabled, this bit is asserted and all DRAM receivers are disabled. The inputs/bidirectional DRAM pins can be left unconnected.

0 = Receivers enabled
1 = Receivers disabled

**Bypass Max (BypMax)**—Bits 27–25. Specifies the number of times the oldest entry in DCQ can be bypassed in arbitration before the arbiter's choice is vetoed. When programmed to all 0s, this feature is disabled, and the choice of the arbiter is always respected.

### 3.5.9.2 DRAM Configuration High Register

**DRAM Configuration High Register** **Function 2: Offset 94h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–30 | reserved | | R | 0 |
| 29 | MC3_EN | Memory Clock 3 Enabled | R/W | 0 |
| 28 | MC2_EN | Memory Clock 2 Enabled | R/W | 0 |
| 27 | MC1_EN | Memory Clock 1 Enabled | R/W | 0 |
| 26 | MC0_EN | Memory Clock 0 Enabled | R/W | 0 |
| 25 | MCR | Memory Clock Ratio Valid | R/W | 0 |
| 24–23 | reserved | | R | 0 |
| 22–20 | MemClk | DRAM MEMCLK Frequency | R/W | 0 |
| 19 | DCC_EN | Dynamic Idle Cycle Center Enable | R/W | 0 |
| 18–16 | ILD_lmt | Idle Cycle Limit | R/W | 0 |
| 15–12 | reserved | | R | 0 |
| 11–8 | RdPreamble | Read Preamble | R/W | 0 |
| 7–4 | reserved | | R | 0 |
| 3–0 | AsyncLat | Maximum Asynchronous Latency | R/W | 0 |

#### Field Descriptions

**Maximum Asynchronous Latency (AsyncLat)**—Bits 3–0. This field should be loaded with a 4-bit value equal to the maximum asynchronous latency in the DRAM read round-trip loop.

0000b = 0 ns
...
1111b = 15 ns

**Read Preamble (RdPreamble)**—Bits 11–8. The time prior to the max-read DQS-return when the DQS receiver should be turned on. This is specified in units of 0.5 ns. The controller needs to know when to enable its DQS receiver in anticipation of the DRAM DQS driver turning on for a read. The controller will disable its DQS receiver until the read preamble time and then enable its DQS receiver while the DRAM asserts DQS.

0000b = 2.0 ns
0001b = 2.5 ns
0010b = 3.0 ns
0011b = 3.5 ns
0100b = 4.0 ns
0101b = 4.5 ns
0110b = 5.0 ns
0111b = 5.5 ns
1000b = 6.0 ns
1001b = 6.5 ns
1010b = 7.0 ns
1011b = 7.5 ns
1100b = 8.0 ns
1101b = 8.5 ns
1110b = 9.0 ns
1111b = 9.5 ns

**Idle Cycle Limit (ILD_lmt)**—Bits 18–16. Specifies the number of MemCLKs before forcibly closing (precharging) an open page. If DCC_EN (Function 2, Offset 94h) has a value of 0, the static counters are loaded with the ILD_lmt and decremented each clock. If DCC_EN (Function 2, Offset 94h) has a value of 1, the dynamic counters are loaded with the ILD_lmt and modified as follows:

Increment—When a Page Miss (PM) page hits on an invalid entry in the Page Table. The presumption is that in the past, that page table entry was occupied by the very same page that has a Page Miss. Had the old page been kept open longer, it would have been a Page Hit. Increment the Idle Cycle Limit count to increase the probability of getting a future Page Hit.

Decrement—When a Page Conflict (PC) arrives and hits on an idle entry (obviously an open page). This Page Conflict can be avoided if the open page is closed earlier. Decrement the Idle Cycle Limit count to increase the probability of avoiding a future Page Conflict.

000b = 0 cycles
001b = 4 cycles
010b = 8 cycles
011b = 16 cycles
100b = 32 cycles
101b = 64 cycles
110b = 128 cycles
111b = 256 cycles

**Dynamic Idle Cycle Center Enable (DCC_EN)**—Bit 19. When set to 1, indicates that each entry in the page table dynamically adjusts the idle cycle limit based on Page Conflict/Page Miss (PC/PM) traffic.

**DRAM MEMCLK Frequency (MemClk)**—Bits 22–20. Specifies the DRAM MEMCLK frequency in MHz. It is possible to program this field for a higher frequency than the maximum allowed by the processor. Refer to the processor data sheet for the maximum operating frequency allowed for a given processor implementation.

000b = 100 MHz
001b = reserved
010b = 133 MHz
011b = reserved
100b = reserved
101b = 166 MHz
110b = reserved
111b = 200 MHz

**Memory Clock Ratio Valid (MCR)**—Bit 25. Indicates to the controller to drive MEMCLK. The following fields need to be programmed before this bit is set: MemClk (Function 2, Offset 94h), MCi_EN (Function 2, Offset 94h), UnBuffDimm (Function 2, Offset 90h), and 128/64 (Function 2, Offset 90h). This bit is also set after the Base Address registers have been configured. This allows the system to know if a CS is occupied before clocks start toggling. (The intent is to not drive clocks to unoccupied slots to reduce the potential for EMI.)

0 = Disable MemCLKs (default)
1 = Enable MemCLKs

**Memory Clock 0 Enabled (MC0_EN)**—Bit 26. The MC0_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM0.

0 = Disabled (default)
1 = Enabled

**Memory Clock 1 Enabled (MC1_EN)**—Bit 27. The MC1_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM1.

0 = Disabled (default)
1 = Enabled

**Memory Clock 2 Enabled (MC2_EN)**—Bit 28. The MC2_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM2.

0 = Disabled (default)
1 = Enabled

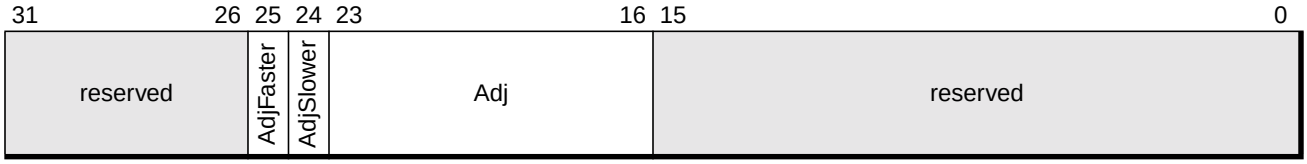
**Memory Clock 3 Enabled (MC3_EN)**—Bit 29. The MC3_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM3.

0 = Disabled (default)
1 = Enabled

### 3.5.10 DRAM Delay Line Register

The DRAM Delay Line register is used to adjust the skew of the input DQS strobe relative to the data.

**DRAM Delay Line Register** **Function 2: Offset 98h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–26 | reserved | | R | 0 |
| 25 | AdjFaster | Adjust Faster | R/W | 0 |
| 24 | AdjSlower | Adjust Slower | R/W | 0 |
| 23–16 | Adj | Delay Line Adjust | R/W | 0 |
| 15–0 | reserved | | R | 0 |

#### Field Descriptions

**Delay Line Adjust (Adj)**—Bits 23–16. Adjusts the DLL derived PDL delay by one or more delay stages in either the faster or slower direction.

**Adjust Slower (AdjSlower)**—Bit 24. Indicates that the value in Adj is used to increase the PDL delay.

**Adjust Faster (AdjFaster)**—Bit 25. Indicates that the value in Adj is used to decrease the PDL delay.

## 3.6 Function 3—Miscellaneous Control

This function is a collection of the remaining memory system configuration registers. As listed in Table 10, Function 3 includes the following registers:

- Machine check architecture for Northbridge errors
- DRAM scrubber
- Buffer counts for Northbridge flow control
- Power management
- GART
- Clocking/FIFO control

- Thermtrip status
- Northbridge capabilities

**Table 10. Function 3 Configuration Registers**

| Offset | Register Name | | | | Reset | Access | Description |
|---|---|---|---|---|---|---|---|
| 00h | Device ID | | Vendor ID (AMD) | | 1103_1022h | RO | page 90 |
| 04h | Status[1] | | Command | | 0000_0000h | RO | |
| 08h | Base Class Code | Subclass Code | Program Interface | Revision ID | 0600_0000h | RO | page 91 |
| 0Ch | BIST | Header Type | Latency Timer | Cache Line Size | 0080_0000h | RO | page 91 |
| 10h | Base Address 0 | | | | 0000_0000h | RO | |
| 14h | Base Address 1 | | | | 0000_0000h | RO | |
| 18h | Base Address 2 | | | | 0000_0000h | RO | |
| 1Ch | Base Address 3 | | | | 0000_0000h | RO | |
| 20h | Base Address 4 | | | | 0000_0000h | RO | |
| 24h | Base Address 5 | | | | 0000_0000h | RO | |
| 28h | Card Bus CIS Pointer | | | | 0000_0000h | RO | |
| 2Ch | Sub System ID | | Sub system Vendor ID | | 0000_0000h | RO | |
| 30h | ROM Base Address | | | | 0000_0000h | RO | |
| 34h | Capabilities | | | | 0000_0000h | RO | |
| 38h | reserved | | | | 0000_0000h | RO | |
| 3Ch | Max Latency | Min GNT | Int Pin | Int Line | 0000_0000h | RO | |
| 40h | MCA Northbridge Control | | | | 0000_0000h | RW | page 92 |
| 44h | MCA Northbridge Configuration | | | | 0000_0000h | RW | page 92 |
| 48h | MCA Northbridge Status Low | | | | page 98 | RW | page 98 |
| 4Ch | MCA Northbridge Status High | | | | page 101 | RW | page 101 |
| 50h | MCA Northbridge Address Low | | | | page 103 | RW | page 103 |
| 54h | MCA Northbridge Address High | | | | page 104 | RW | page 104 |
| 58h | Scrub Control | | | | 0000_0000h | RW | page 110 |
| 5Ch | DRAM Scrub Address Low | | | | page 112 | RW | page 112 |
| 60h | DRAM Scrub Address High | | | | page 113 | RW | page 113 |
| 70h | SRI-to-XBAR Buffer Counts | | | | 5102_0111h | RW | page 116 |

***Notes:***

1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

**Table 10. Function 3 Configuration Registers (Continued)**

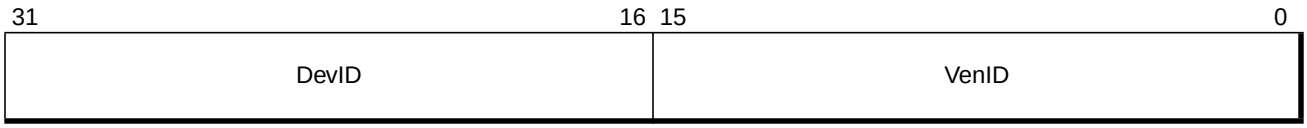
| Offset | Register Name | Reset | Access | Description |
|---|---|---|---|---|
| 74h | XBAR-to-SRI Buffer Counts | 5000_8011h | RW | page 118 |
| 78h | MCT-to-XBAR Buffer Counts | 0800_3800h | RW | page 117 |
| 7Ch | Free List Buffer Counts | 0000_221Bh | RW | page 117 |
| 80h | Power Management Control Low | 0000_0000h | RW | page 121 |
| 84h | Power Management Control High | 0000_0000h | RW | page 122 |
| 90h | GART Aperture Control | 0000_0000h | RW | page 122 |
| 94h | GART Aperture Base | page 124 | RW | page 124 |
| 98h | GART Table Base | page 124 | RW | page 124 |
| 9Ch | GART Cache Control | 0000_0000h | RW | page 125 |
| D4h | Clock Power/Timing Low | page 126 | RW | page 126 |
| D8h | Clock Power/Timing High | page 128 | RW | page 128 |
| DCh | HyperTransport™ Read Pointer Optimization | page 128 | RW | page 128 |
| E4h | Thermtrip Status | 0000_0000h | RO | page 130 |
| E8h | Northbridge Capabilities | 0000_0000h | RO | page 131 |

***Notes:***
1. *The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.*
2. *Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.*

## 3.6.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 3 registers and is part of the standard PCI configuration header.

### Device/Vendor ID Register — Function 3: Offset 00h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | DevID | Device ID | R | 1103h |
| 15–0 | VenID | Vendor ID | R | 1022h |

### Field Descriptions

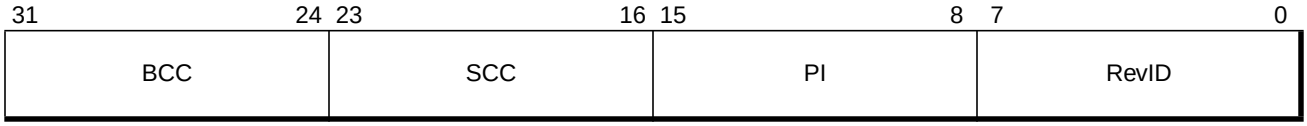
**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1103h for the HyperTransport technology configuration function.

## 3.6.2 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 3 registers and is part of the standard PCI configuration header.

**Class Code/Revision ID Register** **Function 3: Offset 08h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BCC | Base Class Code | R | 06h |
| 23–16 | SCC | Subclass Code | R | 00h |
| 15–8 | PI | Programming Interface | R | 00h |
| 7–0 | RevID | Revision ID | R | 00h |

### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

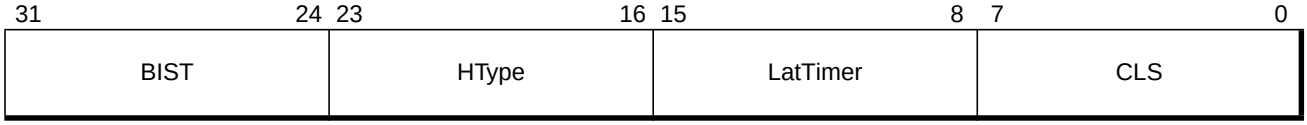
**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

## 3.6.3 Header Type Register

This register specifies the header type for the Function 3 registers and is part of the standard PCI configuration header.

**Header Type Register** **Function 3: Offset 0Ch**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | BIST | BIST | R | 00h |
| 23–16 | HType | Header Type | R | 80h |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 15–8 | LatTimer | Latency Timer | R | 00h |
| 7–0 | CLS | Cache Line Size | R | 00h |

### Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (Htype)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

## 3.6.4 Machine Check Architecture (MCA) Registers

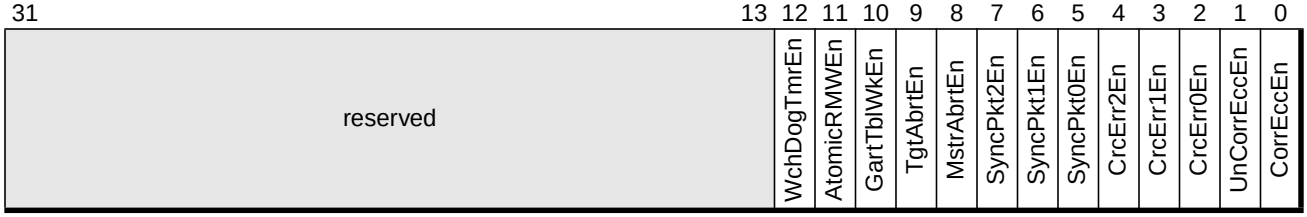
These registers are used to configure the Machine Check Architecture (MCA) functions of the on-chip Northbridge (NB) hardware and to provide a method for the Northbridge hardware to report errors in a way compatible with MCA. All of the Northbridge MCA registers, except for the MCA NB Configuration register, are accessible through the MCA-defined MSR method, as well as through PCI configuration space. The MCA NB Configuration register is accessible only through PCI configuration space.

The MCA NB Control register enables MCA reporting of each error checked by the Northbridge. The global MCA error enables must also be set through the global MCA MSRs. The error enables in this register only affect error reporting through MCA. Actions which the Northbridge may take in addition to MCA reporting are enabled through the MCA NB Configuration register.

The MCA NB Status registers and MCA NB Address registers provide status and address information regarding an error which the Northbridge has logged. The values of error-logging registers are maintained through a warm reset allowing software to identify an error source that resulted in system-wide initialization.

### 3.6.4.1 MCA NB Control Register

**MCA NB Control Register** **Function 3: Offset 40h**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–13 | reserved | | R | 0 |
| 12 | WchDogTmrEn | Watchdog Timer Error Reporting Enable | R/W | 0 |
| 11 | AtomicRMWEn | Atomic Read-Modify-Write Error Reporting Enable | R/W | 0 |
| 10 | GartTblWkEn | GART Table Walk Error Reporting Enable | R/W | 0 |
| 9 | TgtAbrtEn | Target Abort Error Reporting Enable | R/W | 0 |
| 8 | MstrAbrtEn | Master Abort Error Reporting Enable | R/W | 0 |
| 7 | SyncPkt2En | HyperTransport Link 2 Sync Packet Error Reporting Enable | R/W | 0 |
| 6 | SyncPkt1En | HyperTransport Link 1 Sync Packet Error Reporting Enable | R/W | 0 |
| 5 | SyncPkt0En | HyperTransport Link 0 Sync Packet Error Reporting Enable | R/W | 0 |
| 4 | CrcErr2En | HyperTransport Link 2 CRC Error Reporting Enable | R/W | 0 |
| 3 | CrcErr1En | HyperTransport Link 1 CRC Error Reporting Enable | R/W | 0 |
| 2 | CrcErr0En | HyperTransport Link 0 CRC Error Reporting Enable | R/W | 0 |
| 1 | UnCorrEccEn | Uncorrectable ECC Error Reporting Enable | R/W | 0 |
| 0 | CorrEccEn | Correctable ECC Error Reporting Enable | R/W | 0 |

## Field Descriptions

**Correctable ECC Error Reporting Enable (CorrEccEn)**—Bit 0. Enables MCA reporting of correctable ECC errors which are detected in the Northbridge. Since correctable errors do not result in an MCA exception the effect of this bit is limited to the setting of the error enable bit in the MCA NB Status High register.

**Uncorrectable ECC Error Reporting Enable (UnCorrEccEn)**—Bit 1. Enables MCA reporting of uncorrectable ECC errors which are detected in the Northbridge. Note that in some cases data may be forwarded to the CPU core prior to checking ECC in which case the check takes place in one of the other error reporting banks.

**HyperTransport Link 0 CRC Error Reporting Enable (CrcErr0En)**—Bit 2. Enables MCA reporting of CRC errors detected on HyperTransport link 0. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a CRC error on an incoming link independent of the state of this bit.

**HyperTransport Link 1 CRC Error Reporting Enable (CrcErr1En)**—Bit 3. Enables MCA reporting of CRC errors detected on HyperTransport link 1. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a CRC error on an incoming link independent of the state of this bit.

**HyperTransport Link 2 CRC Error Reporting Enable (CrcErr2En)**—Bit 4. Enables MCA reporting of CRC errors detected on HyperTransport link 2. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a CRC error on an incoming link independent of the state of this bit.

**HyperTransport Link 0 Sync Packet Error Reporting Enable (SyncPkt0En)**—Bit 5. Enables MCA reporting of HyperTransport sync error packets detected on HyperTransport link 0. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a sync packet on an incoming link independent of the state of this bit.

**HyperTransport Link 1 Sync Packet Error Reporting Enable (SyncPkt1En)**—Bit 6. Enables MCA reporting of HyperTransport sync error packets detected on HyperTransport link 1. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a sync packet on an incoming link independent of the state of this bit.

**HyperTransport Link 2 Sync Packet Error Reporting Enable (SyncPkt2En)**—Bit 7. Enables MCA reporting of HyperTransport sync error packets detected on HyperTransport link 2. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a sync packet on an incoming link independent of the state of this bit.

**Master Abort Error Reporting Enable (MstrAbrtEn)**—Bit 8. Enables MCA reporting of master aborts (HyperTransport packets that return an error status with the Non Existent Address bit set). The Northbridge will return an error response back to the requestor with any associated data all 1s independent of the state of this bit.

**Target Abort Error Reporting Enable (TgtAbrtEn)**—Bit 9. Enables MCA reporting of target aborts (HyperTransport technology packets that return an error status with the Non Existent Address bit clear). The Northbridge will return an error response back to the requestor with any associated data all 1s independent of the state of this bit.

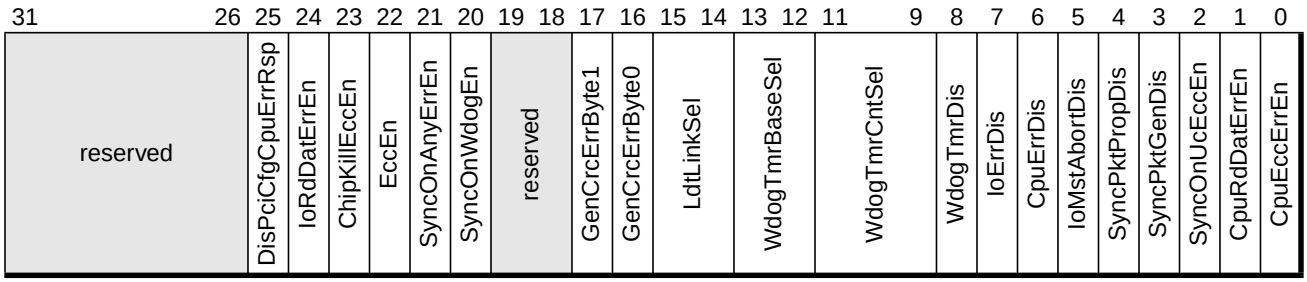
**GART Table Walk Error Reporting Enable (GartTblWkEn)**—Bit 10. Enables MCA reporting of GART cache table walks which encounter a GART PTE entry which is invalid.

**Atomic Read-Modify-Write Error Reporting Enable (AtomicRMWEn)**—Bit 11. Enables MCA reporting of atomic read-modify-write (RMW) HyperTransport technology commands received from an noncoherent HyperTransport link which do not target DRAM. Atomic RMW commands to memory-mapped I/O are not supported in HyperTransport technology. An atomic RMW command that targets MMIO results in a HyperTransport error response being generated back to the requesting I/O device. The generation of the HyperTransport error response is not affected by this bit.

**Watchdog Timer Error Reporting Enable (WchDogTmrEn)**—Bit 12. Enables MCA reporting of watchdog timer errors. The watchdog timer checks for Northbridge system accesses for which a response is expected and where no response is received. See the MCA NB Configuration register for information regarding configuration of the watchdog timer duration. Note that this bit does not affect operation of the watchdog timer in terms of its ability to complete an access that would otherwise cause a system hang. This bit only affects whether such errors are reported through MCA.

### 3.6.4.2 MCA NB Configuration Register

**MCA NB Configuration Register** **Function 3: Offset 44h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–26 | reserved | | R | 0 |
| 25 | DisPciCfgCpuErrRsp | PCI configuration CPU Error Response Disable | R/W | 0 |
| 24 | IoRdDatErrEn | I/O Read Data Error Log Enable | R/W | 0 |
| 23 | ChipKillEccEn | Chip-Kill ECC Mode Enable | R/W | 0 |
| 22 | EccEn | ECC Enable | R/W | 0 |
| 21 | SyncOnAnyErrEn | Sync Flood On Any Error Enable | R/W | 0 |
| 20 | SyncOnWdogEn | Sync Flood on Watchdog Timer Error Enable | R/W | 0 |
| 19–18 | reserved | | R | 0 |
| 17 | GenCrcErrByte1 | Generate CRC Error on Byte Lane 1 | R/W | 0 |
| 16 | GenCrcErrByte0 | Generate CRC Error on Byte Lane 0 | R/W | 0 |
| 15–14 | LdtLinkSel | HyperTransport Link Select for CRC Error Generation | R/W | 0 |
| 13–12 | WdogTmrBaseSel | Watchdog Timer Time Base Select | R/W | 0 |
| 11–9 | WdogTmrCntSel | Watchdog Timer Count Select | R/W | 0 |
| 8 | WdogTmrDis | Watchdog Timer Disable | R/W | 0 |
| 7 | IoErrDis | I/O Error Response Disable | R/W | 0 |
| 6 | CpuErrDis | CPU Error Response Disable | R/W | 0 |
| 5 | IoMstAbortDis | I/O Master Abort Error Response Disable | R/W | 0 |
| 4 | SyncPktPropDis | Sync Packet Propagation Disable | R/W | 0 |
| 3 | SyncPktGenDis | Sync Packet Generation Disable | R/W | 0 |
| 2 | SyncOnUcEccEn | Sync Flood on Uncorrectable ECC Error Enable | R/W | 0 |
| 1 | CpuRdDatErrEn | CPU Read Data Error Log Enable | R/W | 0 |
| 0 | CpuEccErrEn | CPU ECC Error Log Enable | R/W | 0 |

#### Field Descriptions

**CPU ECC Error Log Enable (CpuEccErrEn)**—Bit 0. Enables reporting of ECC errors for data destined for the CPU on this node. This bit should be clear if ECC error logging is enabled for the remaining error reporting blocks in the CPU. Logging the same error in more than one block may cause a single error event to be treated as a multiple error event and cause the CPU to enter shutdown.

**CPU Read Data Error Log Enable (CpuRdDatErrEn)**—Bit 1. Enables reporting of read data errors (master aborts and target aborts) for data destined for the CPU on this node. This bit should be clear if read data error logging is enabled for the remaining error reporting blocks in the CPU. Logging the same error in more than one block may cause a single error event to be treated as a multiple error event and cause the CPU to enter shutdown.

**Sync Flood on Uncorrectable ECC Error Enable (SyncOnUcEccEn)**—Bit 2. Enables flooding of all HyperTransport links with sync packets on detection of an uncorrectable ECC error.

**Sync Packet Generation Disable (SyncPktGenDis)**—Bit 3. Disables flooding of all outgoing HyperTransport links with sync packets when a CRC error is detected on an incoming link. By default, sync packet generation for CRC errors is controlled through the HyperTransport technology LDTn Link Control registers (see page 43).

**Sync Packet Propagation Disable (SyncPktPropDis)**—Bit 4. Disables flooding of all outgoing HyperTransport links with sync packets when a sync packet is detected on an incoming link. Sync packets are propagated by default.

**I/O Master Abort Error Response Disable (IoMstAbortDis)**—Bit 5. Disables setting the NXA bit in HyperTransport response packets to I/O devices on detection of a master abort HyperTransport technology error condition.

**CPU Error Response Disable (CpuErrDis)**—Bit 6. Disables generation of a read data error response to the CPU core on detection of a target or master abort HyperTransport technology error condition.

**I/O Error Response Disable (IoErrDis)**—Bit 7. Disables setting the Error bit in HyperTransport response packets to I/O devices on detection of a target or master abort HyperTransport technology error condition.

**Watchdog Timer Disable (WdogTmrDis)**—Bit 8. Disables the watchdog timer. The watchdog timer is enabled by default and checks for Northbridge system accesses for which a response is expected and where no response is received. If such a condition is detected the outstanding access is completed by generating an error response back to the requestor. An MCA error may also be generated if enabled in the MCA NB Control register.

**Watchdog Timer Count Select (WdogTmrCntSel)**—Bit 11–9. Selects the count used by the watchdog timer. The counter selected by WdogTmrCntSel determines the maximum count value in the time base selected by WdogTmrBaseSel.

000b = 4095 (default)
001b = 2047
010b = 1023
011b = 511
100b = 255
101b = 127
110b = 63
111b = 31

**Watchdog Timer Time Base Select (WdogTmrBaseSel)**—Bit 13–12. Selects the time base used by the watchdog timer. The counter selected by WdogTmrCntSel determines the maximum count value in the time base selected by WdogTmrBaseSel.

00b = 1 ms (default)
01b = 1 µs
10b = 5 ns
11b = reserved

**HyperTransport Link Select for CRC Error Generation (LdtLinkSel)**—Bit 15–14. Selects the HyperTransport link to be used for CRC error injection through GenCrcErrByte1/ GenCrcErrByte0.

00b = HyperTransport link 0
01b = HyperTransport link 1
10b = HyperTransport link 2
11b = undefined

**Generate CRC Error on Byte Lane 0 (GenCrcErrByte0)**—Bit 16. Causes a CRC error to be injected on Byte Lane 0 of the HyperTransport link specified by LdtLinkSel. The data carried by the link is unaffected. This bit is cleared after the error has been generated.

**Generate CRC Error on Byte Lane 1 (GenCrcErrByte1)**—Bit 17. Causes a CRC error to be injected on Byte Lane 1 of the HyperTransport link specified by LdtLinkSel. The data carried by the link is unaffected. This bit is cleared after the error has been generated.

**Sync Flood on Watchdog Timer Error Enable (SyncOnWdogEn)**—Bit 20. Enables flooding of all HyperTransport links with sync packets on detection of a watchdog timer error.

**Sync Flood On Any Error Enable (SyncOnAnyErrEn)**—Bit 21. Enables flooding of all HyperTransport links with sync packets on detection of any MCA error that is uncorrectable.

**ECC Enable (EccEn)**—Bit 22. Enables ECC check/correct mode. This bit must be set in order for any ECC checking/correcting to be enabled for any processor block. If set, ECC will be checked and correctable errors will be corrected irrespective of whether machine check ECC reporting is enabled. See "ECC and Chip Kill Error Checking and Correction" on page 108 for more details.

The hardware will only allow values to be programmed into this field which are consistent with the ECC capabilities of the device as specified in the Northbridge Capabilities register (page 131). Attempts to write values inconsistent with the capabilities will result in this field not being updated.

**Chip-Kill ECC Mode Enable (ChipKillEccEn)**—Bit 23. Enables chip-kill ECC mode. Setting this bit causes ECC checking to be based on a 128/16 data/ECC rather than on a 64/8 data/ECC. Chip-kill ECC can only be enabled in 128-bit DRAM data width mode. It allows correction of an error affecting a single symbol rather than the single bit correction available in 64/8 ECC checking. When this bit is set, EccEn must be set. See "ECC and Chip Kill Error Checking and Correction" on page 108 for more details.

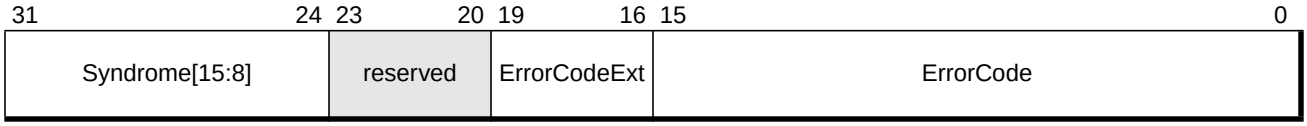
The hardware will only allow values to be programmed into this field which are consistent with the ECC and Chip Kill ECC capabilities of the device as specified in the Northbridge Capabilities register (page 131). Attempts to write values inconsistent with the capabilities will result in this field not being updated.

**I/O Read Data Error Log Enable (IoRdDatErrEn)**—Bit 24. Enables reporting of read data errors (master aborts and target aborts) for data destined for I/O devices. If this bit is clear (default state), master and target aborts for transactions from I/O devices are not logged by MCA, although error responses may still be generated to the requesting I/O device.

**PCI Configuration CPU Error Response Disable (DisPciCfgCpuErrRsp)**—Bit 25. Disables master abort and target abort reporting through the CPU error-reporting banks for PCI configuration accesses. It is recommended that this bit be set in order to avoid MCA exceptions being generated from master aborts for PCI configuration accesses (which can be common during device enumeration).

### 3.6.4.3 MCA NB Status Low Register

**MCA NB Status Low Register** **Function 3: Offset 48h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | Syndrome[15:8] | Syndrome Bits 15–8 for Chip Kill ECC Mode | R/W | X |
| 23–20 | reserved | | R | 0 |
| 19–16 | ErrorCodeExt | Extended Error Code | R/W | X |
| 15–0 | ErrorCode | Error Code | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

#### Field Descriptions

**Error Code (ErrorCode)**—Bits 15–0. Logs an error code when an error is detected. See Table 18 on page 100 for the error codes. Table 11 on page 99 describes the possible error code formats.

**Extended Error Code (ErrorCodeExt)**—Bits 19–16. Logs an extended error code when an error is detected. See Table 18 on page 100 for the extended error codes.

**Syndrome Bits 15–8 for Chip Kill ECC Mode (Syndrome[15–8])**—Bits 31–24. Logs the upper eight syndrome bits when an ECC error is detected. Only valid for ECC errors in Chip-Kill ECC mode.

**Table 11. Error Code Field Formats**

| Error Value | Error Type | Description |
|---|---|---|
| 0000 0000 0001 TTLL | TLB errors | Errors in the GART TLB cache.<br>TT = Transaction Type (Table 12 on page 99)<br>LL = Cache Level (Table 13 on page 99) |
| 0000 0001 RRRR TTLL | Memory errors | Errors in thc cache hierarchy (not applicable for Northbridge errors)<br>RRRR = Memory Transaction Type (Table 14 on page 99)<br>TT = Transaction Type (Table 12 on page 99)<br>LL = Cache Level (Table 13 on page 99) |
| 0000 1PPT RRRR IILL | Bus errors | General bus errors including errors in the HyperTransport™ link or DRAM.<br>PP = Participation Processor (Table 15 on page 100)<br>T = Time-out (Table 16 on page 100)<br>RRRR = Memory Transaction Type (Table 14 on page 99)<br>II = Memory or I/O (Table 17 on page 100)<br>LL = Cache Level (Table 13 on page 99) |

**Table 12. Transaction Type Bits (TT)**

| | |
|---|---|
| 00b | Instruction |
| 01b | Data |
| 10b | Generic |
| 11b | reserved |

**Table 13. Cache Level Bits (LL)**

| | |
|---|---|
| 00b | Level 0 (L0) |
| 01b | Level 1 (L1) |
| 10b | Level 2 (L2) |
| 11b | Generic (LG) |

**Table 14. Memory Transaction Type Bits (RRRR)**

| | |
|---|---|
| 0000b | Generic error (GEN) |
| 0001b | Generic read (RD) |
| 0010b | Generic write (WR) |
| 0011b | Data read |
| 0100b | Data write |
| 0101b | Instruction fetch |

**Table 14. Memory Transaction Type Bits (RRRR) (Continued)**

| | |
|---|---|
| 0110b | Prefetch |
| 0111b | Evict |
| 1000b | Snoop (probe) |

**Table 15. Participation Processor Bits (PP)**

| | |
|---|---|
| 00b | Local node originated the request (SRC) |
| 01b | Local node responded to the request (RES) |
| 10b | Local node observed error as 3rd party (OBS) |
| 11b | Generic |

**Table 16. Time-Out Bit (T)**

| | |
|---|---|
| 0 | Request did not time out |
| 1 | Request timed out (TIMOUT) |

**Table 17. Memory or I/O Bits (II)**

| | |
|---|---|
| 00b | Memory access (MEM) |
| 01b | reserved |
| 10b | I/O access |
| 11b | Generic (GEN) |

Table 18 on page 100 lists the errors reported by the Northbridge, along with the error codes for each error. The error code fields are defined in the table above. See the NB Control register for more information on errors detected by the Northbridge.

**Table 18. Northbridge Error Codes**

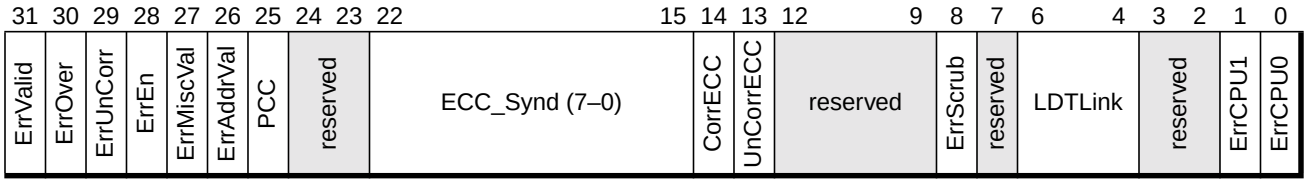
| Error Type | Extended Error Code | Error Code | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Type | PP | T | RRRR | II | LL | TT |
| ECC error | 0000 | BUS | SRC/RSP | 0 | RD/WR | MEM | LG | - |
| CRC error | 0001 | BUS | OBS | 0 | GEN | GEN | GEN | - |
| Sync error | 0010 | BUS | OBS | 0 | GEN | GEN | GEN | - |
| Mst Abort | 0011 | BUS | SRC/OBS | 0 | RD/WR | MEM/IO | GEN | - |
| Tgt Abort | 0100 | BUS | SRC/OBS | 0 | RD/WR | MEM/IO | GEN | - |
| GART error | 0101 | TLB | - | - | - | - | GEN | GEN |
| RMW error | 0110 | BUS | OBS | 0 | GEN | IO | GEN | - |
| Wdog error | 0111 | BUS | GEN | 1 | GEN | GEN | GEN | - |

**Table 18. Northbridge Error Codes**

| Error Type | Extended Error Code | Error Code | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Type | PP | T | RRRR | II | LL | TT |
| ChipKill ECC error | 1000 | BUS | SRC/RSP | 0 | RD/WR | MEM | LG | - |

### 3.6.4.4 MCA NB Status High Register

**MCA NB Status High Register** **Function 3: Offset 4Ch**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31 | ErrValid | Error Valid | R/W | X |
| 30 | ErrOver | Error Overflow | R/W | X |
| 29 | ErrUnCorr | Error Uncorrected | R/W | X |
| 28 | ErrEn | Error Enable | R/W | X |
| 27 | ErrMiscVal | Miscellaneous Error Register Valid | R | X |
| 26 | ErrAddrVal | Error Address Valid | R/W | X |
| 25 | PCC | Processor Context Corrupt | R/W | X |
| 24–23 | reserved | | R | 0 |
| 22–15 | ECC_Synd (7–0) | Syndrome Bits (7–0) for ECC Errors | R/W | X |
| 14 | CorrECC | Correctable ECC Error | R/W | X |
| 13 | UnCorrECC | Uncorrectable ECC Error | R/W | X |
| 12–9 | reserved | | R | 0 |
| 8 | ErrScrub | Error Found by DRAM Scrubber | R/W | X |
| 7 | reserved | | R | 0 |
| 6–4 | LDTLink | HyperTransport Link Number | R/W | X |
| 3–2 | reserved | | R | 0 |
| 1 | ErrCPU1 | Error Associated with CPU 1 | R/W | X |
| 0 | ErrCPU0 | Error Associated with CPU 0 | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

#### Field Descriptions

**Error Associated with CPU 0 (ErrCPU0)**—Bit 0. If set to 1, this bit indicates that the error was associated with CPU 0.

**Error Associated with CPU 1 (ErrCPU1)**—Bit 1. If set to 1, this bit indicates that the error was associated with CPU 1.

**HyperTransport Link Number (LDTLink[2:0])**—Bits 6–4. For errors associated with a HyperTransport link (e.g., CRC errors), this field indicates which link was associated with the error.

LDTLink[0] = Error associated with HyperTransport link 0
LDTLink[1] = Error associated with HyperTransport link 1
LDTLink[2] = Error associated with HyperTransport link 2

**Error Found by DRAM Scrubber (ErrScrub)**—Bit 8. If set to 1, this bit indicates that the error was found by the DRAM scrubber.

**Uncorrectable ECC Error (UnCorrECC)**—Bit 13. If set to 1, this bit indicates that the error was an uncorrectable ECC error.

**Correctable ECC Error (CorrECC)**—Bit 14. If set to 1, this bit indicates that the error was a correctable ECC error.

**Syndrome Bits 7–0 for ECC Errors (ECC_Synd[7:0])**—Bits 22–15. Logs the lower eight syndrome bits when an ECC error is detected.

**Processor Context Corrupt (PCC)**—Bit 25. If set to 1, this bit indicates that the state of the processor may be corrupted by the error condition. Reliable restarting might not be possible.

0 = Processor not corrupted
1 = Processor may be corrupted

**Error Address Valid (ErrAddrVal)**—Bit 26. If set to 1, this bit indicates that the address saved in the address register is the address where the error occurred (i.e. it validates the address in the address register).

0 = Address register not valid
1 = Address register valid

**Miscellaneous Error Register Valid (ErrMiscVal)**—Bit 27. If set to 1, this bit indicates whether the Miscellaneous Error register contains valid information for this error. This bit is read-only 0, since the Miscellaneous Error register is not implemented.

**Error Enable (ErrEn)**—Bit 28. If set to 1, this bit indicates that MCA error reporting is enabled in the MCA Control register.

0 = MCA error reporting not enabled
1 = MCA error reporting enabled

**Error Uncorrected (ErrUnCorr)**—Bit 29. If set to 1, this bit indicates that the error was not corrected by hardware.

0 = Error corrected
1 = Error not corrected

**Error Overflow (ErrOver)**—Bit 30. Set to 1 if the Northbridge detects an error with the valid bit of this register already set. Enabled errors are written over disabled errors, uncorrectable errors are written over correctable errors. Uncorrectable errors are not overwritten.

0 = No error overflow
1 = Error overflow

**Error Valid (ErrValid)**—Bit 31. If set to 1, this bit indicates that a valid error has been detected. This bit should be cleared to 0 by software after the register is read.

0 = No valid error detected
1 = Valid error detected

Table 19 lists the errors reported by Northbridge along with the status bits logged for each error. The status bits are defined in the table above. See the Northbridge Control register for more information on errors detected by Northbridge.

**Table 19. Northbridge Error Status Bit Settings**

| Error Type | ErrUnCorr | ErrAddr Val | PCC | ECC_Synd Valid | Corr ECC | UnCorr ECC | Err Scrub | LDTLink Valid | Err CPU |
|---|---|---|---|---|---|---|---|---|---|
| ECC error | If multi-bit | 1 | If multi-bit and src CPU | 1 | If single-bit | If multi-bit | 1/0 | 0 | 1/0 |
| CRC error | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| Sync error | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| Mst Abort | 1 | 1 | If src CPU | 0 | 0 | 0 | 0 | 1 | 1/0 |
| Tgt Abort | 1 | 1 | If src CPU | 0 | 0 | 0 | 0 | 1 | 1/0 |
| GART error | 1 | 1 | If src CPU | 0 | 0 | 0 | 0 | 0 | 1/0 |
| RMW error | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Wdog error | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | X |
| ChipKill ECC error | If multi-symbol | 1 | If multi-symbol and src CPU | 1 | If single-symbol | If multi-symbol | 1/0 | 0 | 1/0 |

### 3.6.4.5 MCA NB Address Low Register

**MCA NB Address Low Register** **Function 3: Offset 50h**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–3 | ErrAddrLo | Error Address Bits 31–3 | R/W | X |
| 2–0 | reserved | | R | 0 |

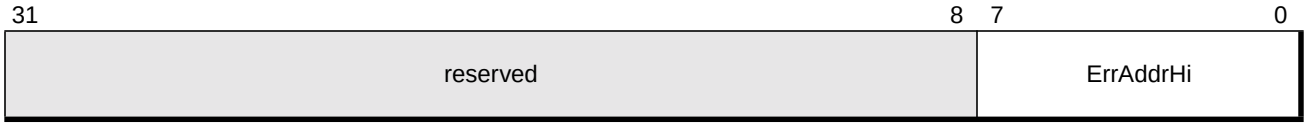
"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Error Address Bits 31–3 (ErrAddrLo)**—Bits 31–3. This field specifies bits 31–3 of the address associated with a machine check error.

### 3.6.4.6 MCA NB Address High Register

**MCA NB Address High Register** **Function 3: Offset 54h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | R | 0 |
| 7–0 | ErrAddrHi | Error Address Bits 39–32 | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

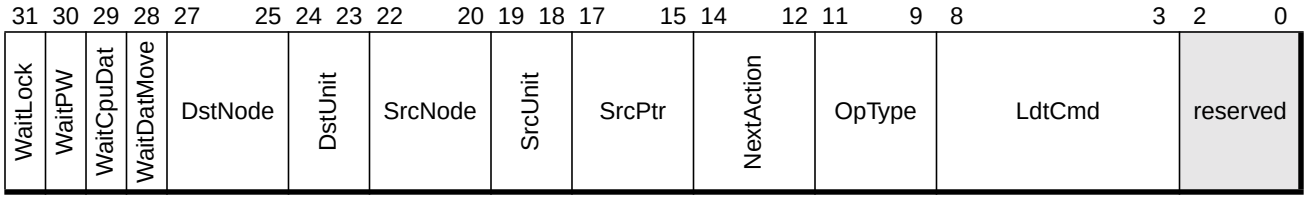
**Error Address Bits 39–32 (ErrAddrHi)**—Bits 7–0. This field specifies bits 39–32 of the address associated with a machine check error.

### 3.6.4.7 Watchdog Timer Errors

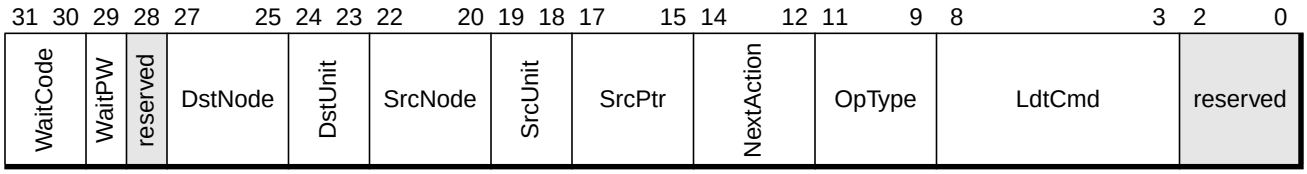
For watchdog timer errors the ErrAddrHi and ErrAddrLo fields in the MCA NB Address High/Low registers log the hardware state information of the request for which the response timed out.

**MCA NB Address Low Register for Watchdog Timer Error** **Function 3: Offset 50h**

Revision B and earlier revisions.




Revision C.




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| Bits [31:28] in revision B and earlier revisions. | | | | |
| 31 | WaitLock | Wait For Bus Lock | | |
| 30 | WaitPW | Wait For Posted Write | | |
| 29 | WaitCpuDat | Wait For CPU Write Data | | |
| 28 | WaitDatMove | Wait For Data Movement | | |
| Bits [31:28] in revision C. | | | | |
| 31-30 | WaitCode | Wait Code (Bits 1-0 of WaitCode) | | |
| 29 | WaitPW | Wait For Posted Write | | |
| 28 | reserved | | | |
| 27–25 | DstNode | Destination Node ID | | |
| 24–23 | DstUnit | Destination Unit ID | | |
| 22–20 | SrcNode | Source Node ID | | |
| 19–18 | SrcUnit | Source Unit ID | | |
| 17–15 | SrcPtr | Source Pointer | | |
| 14–12 | NextAction | Next Action | | |
| 11–9 | OpType | Operation Type | | |
| 8–3 | LdtCmd | HyperTransport Command | | |
| 2–0 | reserved | | | |

## Field Descriptions

**HyperTransport Command (LdtCmd)**—Bit 8–3. The initial command (in HyperTransport technology format) for the stalled operation.

**Operation Type (OpType)**—Bit 11–9. Indicates what type of operation is stalled.

000b = Normal operation (i.e., none of the other types listed)
001b = Bus lock
010b = Local APIC access
011b = Interrupt request
100b = System management request
101b = Interrupt broadcast
110b = Stop grant
111b = SMI ack

**Next Action (NextAction)**—Bit 14–12. Indicates what the stalled operation would have done next had it not become stalled.

000b = Complete
001b = reserved
010b = Send request
011b = Send second request (if any)
100b = Send final response back to requestor
101b = Send initial response (if any) back to requestor
110b = Send final response to the memory controller
111b = Send initial response (if any) to mem controller

**Source Pointer (SrcPtr)**—Bit 17–15. Source pointer of the stalled operation.

000b = Host bridge on local node
001b = CPU on local node
010b = Mem controller on local node
101b = HyperTransport link 0
110b = HyperTransport link 1
111b = HyperTransport link 2

**Source Unit ID (SrcUnit)**—Bit 19–18. Source Unit ID of the stalled operation.

**Source Node ID (SrcNode)**—Bit 22–20. Source Node ID of the stalled operation.

**Destination Unit ID (DstUnit)**—Bit 24–23. Destination Unit ID of the stalled operation.

**Destination Node ID (DstNode)**—Bit 27–25. Destination Node ID of the stalled operation.

Bits [31:28] in revision B and earlier revisions:

**Wait For Data Movement (WaitDatMove)**—Bit 28. When set, indicates that the stalled operation is waiting for read or write data to be moved.

**Wait For CPU Write Data (WaitCpuDat)**—Bit 29. When set, indicates that the stalled operation is waiting for write data from the CPU.

**Wait For Posted Write (WaitPW)**—Bit 30. When set, indicates that a response for the stalled operation is waiting for one or more prior posted writes to complete.

**Wait For Bus Lock (WaitLock)**—Bit 31. When set, indicates that the stalled operation is waiting for a bus lock to complete.

Bits [31:28] in revision C:

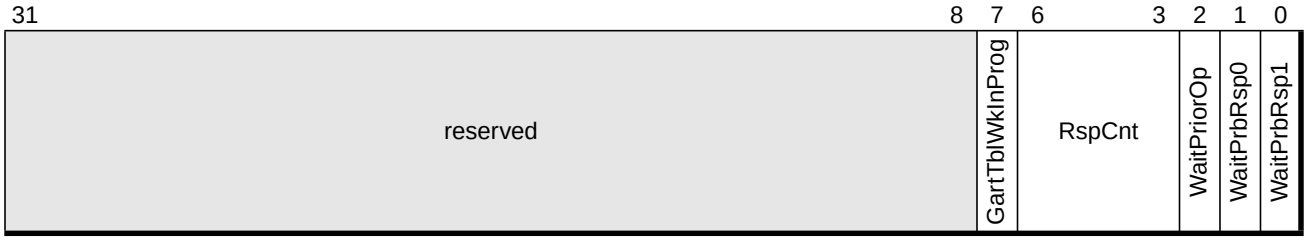
**Wait For Posted Write (WaitPW)**—Bit 29. When set, indicates that a response for the stalled operation is waiting for one or more prior posted writes to complete.

**Wait Code (WaitCode)**—Bit 31-30. WaitCode is a 5 bit field; WaitCode[1:0] is in Function 3, Offset 50h; WaitCode[4:2] is in Function 3, Offset 54h. WaitCode encodings are implementation specific. They describe the reason the hardware is waiting. All zeros mean that there is not a waiting condition. Revision B and earlier revision fields: WaitDatMove, WaitCpuDat, WaitLock (Function 3, Offset 50h), WaitPrbRsp0, WaitPrbRsp1, and WaitPriorOp (Function

3, Offset 50h) are replaced with a code in the revision C field WaitCode (Function 3, Offset 50h, 54h).

### MCA NB Address High Register for Watchdog Timer Error — Function 3: Offset 54h

Revision B and earlier revisions.




Revision C.




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | | |
| 7 | GartTblWkInProg | GART Table Walk In Progress (Bit 39 of ErrAddr) | | |
| 6–3 | RspCnt | System Response Count (Bits 38–35 of ErrAddr) | | |
| Bits [2:0] in revision B and earlier revisions. | | | | |
| 2 | WaitPriorOp | Wait For Prior Operation (Bit 34 of ErrAddr) | | |
| 1 | WaitPrbRsp0 | Wait For CPU0 Probe Response (Bit 33 of ErrAddr) | | |
| 0 | WaitPrbRsp1 | Wait For CPU1 Probe Response (Bit 32 of ErrAddr) | | |
| Bits [2:0] in revision C. | | | | |
| 2-0 | WaitCode | Wait Code (Bits 4-2 of WaitCode) | | |

### Field Descriptions

Bits [2:0] in revision B and earlier revisions:

**Wait For CPU1 Probe Response (WaitPrbRsp1)**—Bit 0 (Bit 32 of ErrAddr). When set, indicates that the stalled operation is waiting on a probe response from CPU1.

**Wait For CPU0 Probe Response (WaitPrbRsp0)**—Bit 1 (Bit 33 of ErrAddr). When set, indicates that the stalled operation is waiting on a probe response from CPU0.

**Wait For Prior Operation (WaitPriorOp)**—Bit 2 (Bit 34 of ErrAddr). When set, indicates that the stalled operation is waiting for a prior operation to release system resources.

Bits [2:0] in revision C:

**Wait Code (WaitCode)**—Bit 2-0. See WaitCode definition in MCA NB Address Low Register for Watchdog Timer Error (Function 3, Offset 50h).

**System Response Count (RspCnt)**—Bits 6–3 (Bits 38–35 of ErrAddr). Number of outstanding system responses for which the stalled operation is waiting.

**GART Table Walk In Progress (GartTblWkInProg)**—Bit 7 (Bit 39 of ErrAddr). Indicates that a GART table walk was in progress at the time the error was logged.

## 3.6.5 ECC and Chip Kill Error Checking and Correction

The memory controller implements two ECC modes: normal ECC and Chip Kill ECC. These error checking and correction modes can only be used if all DIMMs are ECC capable. Chip Kill ECC is only available on the AMD Opteron™ processor.

### 3.6.5.1 Normal ECC

Normal ECC mode is a 64/8 SEC/DED (Single Error Correction/Double Error Detection) Hamming code. It can detect one bit error and correct it, it can detect two bit errors but it can not correct them, and it may detect more than two bit errors depending on the position of corrupted bits. Normal ECC mode is enabled when EccEn, Function 3, Offset 44h is set and ChipKillEccEn, Function 3, Offset 44h is clear.

Error address (Function 3, Offsets 50h and 54h) is valid and uniquely identifies the DIMM with a correctable or uncorrectable error for 64-bit and 128-bit data width configurations. In a 128-bit data width configuration, ECC is independently applied to the lower 64 and upper 64 data bits. In this configuration ECC may detect and correct two bit errors if one error happens on data bits 63-0 and one error happens on data bits 127-64. Status registers (Function 3, Offsets 48h and 4Ch) and error address registers will contain information about the second detected error.

Bit position of a correctable error can be determined by matching Ecc_Synd, Function 3, Offset 4Ch with a value from Table 20.

**Table 20. ECC Syndromes**

| | n = 0 | n = 1 | n = 2 | n = 3 | n = 4 | n = 5 | n = 6 | n = 7 |
|---|---|---|---|---|---|---|---|---|
| Bit (0 + n) | ce | cb | d3 | d5 | d6 | d9 | da | dc |
| Bit (8 +n) | 23 | 25 | 26 | 29 | 2a | 2c | 31 | 34 |
| Bit (16+ n) | 0e | 0b | 13 | 15 | 16 | 19 | 1a | 1c |
| Bit (24+ n) | e3 | e5 | e6 | e9 | ea | ec | f1 | f4 |
| Bit (32+ n) | 4f | 4a | 52 | 54 | 57 | 58 | 5b | 5d |
| Bit (40+ n) | a2 | a4 | a7 | a8 | ab | ad | b0 | b5 |
| Bit (48+ n) | 8f | 8a | 92 | 94 | 97 | 98 | 9b | 9d |

**Table 20. ECC Syndromes**

| | n = 0 | n = 1 | n = 2 | n = 3 | n = 4 | n = 5 | n = 6 | n = 7 |
|---|---|---|---|---|---|---|---|---|
| Bit (56+ n) | 62 | 64 | 67 | 68 | 6b | 6d | 70 | 75 |

### 3.6.5.2 Chip Kill

Chip Kill ECC mode is a 128/16 SSC/DSD (Single Symbol Correction/Double Symbol Detection) BCH code. A symbol is a group of 4 bits which are 4 bit aligned (bits 0–3 make symbol 0, bits 4–7 make symbol 1, etc.). A single symbol error is any bit error combination within one symbol (1 to 4 bits can be corrupted). Chip Kill ECC mode can detect one symbol error and correct it, it can detect two symbol errors but it can not correct them, and it may detect more than two symbol errors depending on the position of corrupted symbols. Chip Kill ECC mode is enabled when EccEn and ChipKillEccEn, Function 3, Offset 44h are set.

Chip Kill mode can only be used in a 128-bit data width configuration.

A DIMM with a correctable error is uniquely identified with error address (Function 3, Offsets 50h and 54h) and Chip Kill syndrome (Syndrome, Function 3, Offset 48h and Ecc_Synd, Function 3, Offset 4Ch). Error address identifies two DIMMs and Chip Kill syndrome identifies one of them. Chip Kill syndromes for symbols 00h-0fh, 20h, and 21h map to data bits 63-0 and Chip Kill syndromes for symbols 10h-1fh, 22h, and 23h map to data bits 127-64 (see Table 21). Corrupted bit positions within a symbol are determined from the Chip Kill syndrome column number in Table 21. Bits set in the column number identify corrupted bits within a symbol. For example, if 6913h is a Chip Kill syndrome, symbol 05h has an error, and bits 0 and 1 within that symbol are corrupted, since the syndrome is in column 3h. Symbol 05h maps to bits 23-20, so the corrupted bits are 20 and 21.

A DIMM with an uncorrectable error can not be uniquely identified. The error address is valid and maps to two DIMMs.

Chip Kill mode can be used with any device width. Chip Kill mode can correct all errors in an x4 device, since the device width is one symbol. Chip Kill mode can correct a subset of errors in an x8 or x16 device.

**Table 21. Chip Kill ECC Syndromes**

| Symbol | 1h | 2h | 3h | 4h | 5h | 6h | 7h | 8h | 9h | ah | bh | ch | dh | eh | fh |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00h | e821 | 7c32 | 9413 | bb44 | 5365 | c776 | 2f57 | dd88 | 35a9 | a1ba | 499b | 66cc | 8eed | 1afe | f2df |
| 01h | 5d31 | a612 | fb23 | 9584 | c8b5 | 3396 | 6ea7 | eac8 | b7f9 | 4cda | 11eb | 7f4c | 227d | d95e | 846f |
| 02h | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 | 0008 | 0009 | 000a | 000b | 000c | 000d | 000e | 000f |
| 03h | 2021 | 3032 | 1013 | 4044 | 6065 | 7076 | 5057 | 8088 | a0a9 | b0ba | 909b | c0cc | e0ed | f0fe | d0df |
| 04h | 5041 | a082 | f0c3 | 9054 | c015 | 30d6 | 6097 | e0a8 | b0e9 | 402a | 106b | 70fc | 20bd | d07e | 803f |
| 05h | be21 | d732 | 6913 | 2144 | 9f65 | f676 | 4857 | 3288 | 8ca9 | e5ba | 5b9b | 13cc | aded | c4fe | 7adf |
| 06h | 4951 | 8ea2 | c7f3 | 5394 | 1ac5 | dd36 | 9467 | a1e8 | e8b9 | 2f4a | 661b | f27c | bb2d | 7cde | 358f |

**Table 21. Chip Kill ECC Syndromes**

| Symbol | 1h | 2h | 3h | 4h | 5h | 6h | 7h | 8h | 9h | ah | bh | ch | dh | eh | fh |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 07h | 74e1 | 9872 | ec93 | d6b4 | a255 | 4ec6 | 3a27 | 6bd8 | 1f39 | f3aa | 874b | bd6c | c98d | 251e | 51ff |
| 08h | 15c1 | 2a42 | 3f83 | cef4 | db35 | e4b6 | f177 | 4758 | 5299 | 6d1a | 78db | 89ac | 9c6d | a3ee | b62f |
| 09h | 3d01 | 1602 | 2b03 | 8504 | b805 | 9306 | ae07 | ca08 | f709 | dc0a | e10b | 4f0c | 720d | 590e | 640f |
| 0ah | 9801 | ec02 | 7403 | 6b04 | f305 | 8706 | 1f07 | bd08 | 2509 | 510a | c90b | d60c | 4e0d | 3a0e | a20f |
| 0bh | d131 | 6212 | b323 | 3884 | e9b5 | 5a96 | 8ba7 | 1cc8 | cdf9 | 7eda | afeb | 244c | f57d | 465e | 976f |
| 0ch | e1d1 | 7262 | 93b3 | b834 | 59e5 | ca56 | 2b87 | dc18 | 3dc9 | ae7a | 4fab | 642c | 85fd | 164e | f79f |
| 0dh | 6051 | b0a2 | d0f3 | 1094 | 70c5 | a036 | c067 | 20e8 | 40b9 | 904a | f01b | 307c | 502d | 80de | e08f |
| 0eh | a4c1 | f842 | 5c83 | e6f4 | 4235 | 1eb6 | ba77 | 7b58 | df99 | 831a | 27db | 9dac | 396d | 65ee | c12f |
| 0fh | 11c1 | 2242 | 3383 | c8f4 | d935 | eab6 | fb77 | 4c58 | 5d99 | 6e1a | 7fdb | 84ac | 956d | a6ee | b72f |
| 10h | 45d1 | 8a62 | cfb3 | 5e34 | 1be5 | d456 | 9187 | a718 | e2c9 | 2d7a | 68ab | f92c | bcfd | 734e | 369f |
| 11h | 63e1 | b172 | d293 | 14b4 | 7755 | a5c6 | c627 | 28d8 | 4b39 | 99aa | fa4b | 3c6c | 5f8d | 8d1e | eeff |
| 12h | b741 | d982 | 6ec3 | 2254 | 9515 | fbd6 | 4c97 | 33a8 | 84e9 | ea2a | 5d6b | 11fc | a6bd | c87e | 7f3f |
| 13h | dd41 | 6682 | bbc3 | 3554 | e815 | 53d6 | 8e97 | 1aa8 | c7e9 | 7c2a | a16b | 2ffc | f2bd | 497e | 943f |
| 14h | 2bd1 | 3d62 | 16b3 | 4f34 | 64e5 | 7256 | 5987 | 8518 | aec9 | b87a | 93ab | ca2c | e1fd | f74e | dc9f |
| 15h | 83c1 | c142 | 4283 | a4f4 | 2735 | 65b6 | e677 | f858 | 7b99 | 391a | badb | 5cac | df6d | 9dee | 1e2f |
| 16h | 8fd1 | c562 | 4ab3 | a934 | 26e5 | 6c56 | e387 | fe18 | 71c9 | 3b7a | b4ab | 572c | d8fd | 924e | 1d9f |
| 17h | 4791 | 89e2 | ce73 | 5264 | 15f5 | db86 | 9c17 | a3b8 | e429 | 2a5a | 6dcb | f1dc | b64d | 783e | 3faf |
| 18h | 5781 | a9c2 | fe43 | 92a4 | c525 | 3b66 | 6ce7 | e3f8 | b479 | 4a3a | 1dbb | 715c | 26dd | d89e | 8f1f |
| 19h | bf41 | d582 | 6ac3 | 2954 | 9615 | fcd6 | 4397 | 3ea8 | 81e9 | eb2a | 546b | 17fc | a8bd | c27e | 7d3f |
| 1ah | 9391 | e1e2 | 7273 | 6464 | f7f5 | 8586 | 1617 | b8b8 | 2b29 | 595a | cacb | dcdc | 4f4d | 3d3e | aeaf |
| 1bh | cce1 | 4472 | 8893 | fdb4 | 3155 | b9c6 | 7527 | 56d8 | 9a39 | 12aa | de4b | ab6c | 678d | ef1e | 23ff |
| 1ch | a761 | f9b2 | 5ed3 | e214 | 4575 | 1ba6 | bcc7 | 7328 | d449 | 8a9a | 2dfb | 913c | 365d | 688e | cfef |
| 1dh | ff61 | 55b2 | aad3 | 7914 | 8675 | 2ca6 | d3c7 | 9e28 | 6149 | cb9a | 34fb | e73c | 185d | b28e | 4def |
| 1eh | 5451 | a8a2 | fcf3 | 9694 | c2c5 | 3e36 | 6a67 | ebe8 | bfb9 | 434a | 171b | 7d7c | 292d | d5de | 818f |
| 1fh | 6fc1 | b542 | da83 | 19f4 | 7635 | acb6 | c377 | 2e58 | 4199 | 9b1a | f4db | 37ac | 586d | 82ee | ed2f |
| 20h | be01 | d702 | 6903 | 2104 | 9f05 | f606 | 4807 | 3208 | 8c09 | e50a | 5b0b | 130c | ad0d | c40e | 7a0f |
| 21h | 4101 | 8202 | c303 | 5804 | 1905 | da06 | 9b07 | ac08 | ed09 | 2e0a | 6f0b | f40c | b50d | 760e | 370f |
| 22h | c441 | 4882 | 8cc3 | f654 | 3215 | bed6 | 7a97 | 5ba8 | 9fe9 | 132a | d76b | adfc | 69bd | e57e | 213f |
| 23h | 7621 | 9b32 | ed13 | da44 | ac65 | 4176 | 3757 | 6f88 | 19a9 | f4ba | 829b | b5cc | c3ed | 2efe | 58df |

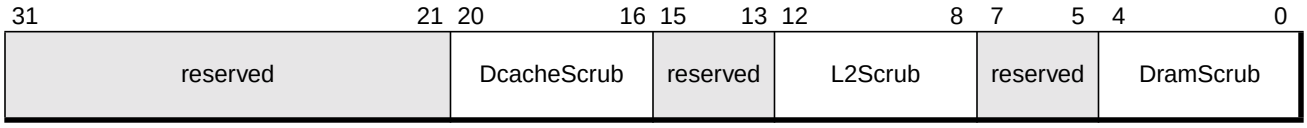
## 3.6.6 Scrub Control Register

This register specifies the scrub rate for sequential ECC scrubbing of DRAM, the L2 cache and the DCACHE. The scrub rate specifies the duration between successive scrub events. A value of 0 disables scrubbing.

Each scrub event corresponds to:

- 64 bytes for the DRAM scrubber.
- 64 bits for the data cache scrubber.
- One single L2 cache line tag address for the L2 scrubber.

### Scrub Control Register — Function 3: Offset 58h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–21 | reserved | | R | 0 |
| 20–16 | DcacheScrub | Data Cache Scrub Rate | R/W | 0 |
| 15–13 | reserved | | R | 0 |
| 12–8 | L2Scrub | L2 Cache Scrub Rate | R/W | 0 |
| 7–5 | reserved | | R | 0 |
| 4–0 | DramScrub | DRAM Scrub Rate | R/W | 0 |

### Field Descriptions

**DRAM Scrub Rate (DramScrub)**—Bits 4–0. Specifies the scrub rate for the DRAM. See Table 22. For example, if 256MByte memory is scrubbed every 12 hours, a 64-byte memory block should be scrubbed every 10ms, and scrubbing rate of 10.49ms should be selected.

**L2 Cache Scrub Rate (L2Scrub)**—Bits 12–8. Specifies the scrub rate for the L2 cache. See Table 22.

**Data Cache Scrub Rate (DcacheScrub)**—Bits 20–16. Specifies the scrub rate for the data cache. See Table 22.

**Table 22. Scrub Rate Control Values**

| Scrub Rate Code | Scrub Rate | Scrub Rate Code | Scrub Rate |
|---|---|---|---|
| 00000b | Do not scrub | 01100b | 81.9 µs |
| 00001b | 40.0 ns | 01101b | 163.8 µs |
| 00010b | 80.0 ns | 01110b | 327.7 µs |
| 00011b | 160.0 ns | 01111b | 655.4 µs |
| 00100b | 320.0 ns | 10000b | 1.31 ms |
| 00101b | 640.0 ns | 10001b | 2.62 ms |
| 00110b | 1.28 µs | 10010b | 5.24 ms |
| 00111b | 2.56 µs | 10011b | 10.49 ms |

**Table 22. Scrub Rate Control Values (Continued)**

| Scrub Rate Code | Scrub Rate | Scrub Rate Code | Scrub Rate |
|---|---|---|---|
| 01000b | 5.12 µs | 10100b | 20.97 ms |
| 01001b | 10.2 µs | 10101b | 42.00 ms |
| 01010b | 20.5 µs | 10110b | 84.00 ms |
| 01011b | 41.0 µs | All others | reserved |

## 3.6.7 DRAM Scrub Address Registers

These registers specify the next address to be scrubbed by the DRAM scrubber. They should be initialized by the BIOS before the scrubber is enabled (the DRAM scrubber is enabled by writing a valid scrub rate to DramScrub field, Function 3, Offset 58h) and after MemClrStatus, Function 2, Offset 90h is set. They are then updated by the scrubber hardware as it scrubs successive 64-byte blocks of DRAM. Once the scrubber reaches the DRAM limit address for the node (see "DRAM Address Map" on page 57), it wraps to the DRAM base address.

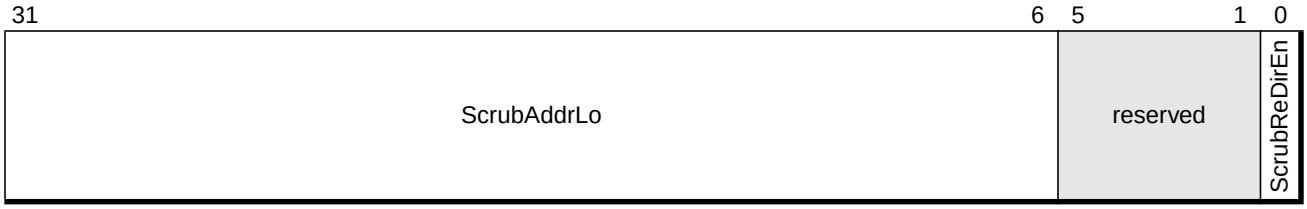
The initial scrubbing address specified by these registers must be between the base and limit address for the node, defined through the DRAM address maps (see "DRAM Address Map" on page 57). The recommended initial scrubbing address is the DRAM base address.

In addition to sequential DRAM scrubbing, the DRAM scrubber has a redirect mode for scrubbing DRAM locations accessed during normal operation. This is enabled by setting ScrubReDirEn (Function 3, Offset 5Ch). When a DRAM read is generated by any agent other than the DRAM scrubber, correctable ECC errors are corrected as the data is passed to the requestor, but the data in DRAM is not corrected if redirect scrubbing mode is disabled. In scrubber redirect mode, correctable errors detected during normal DRAM read accesses redirect the scrubber to the location of the error. After the scrubber corrects the location in DRAM, it resumes scrubbing from where it left off. DRAM scrub address registers are not modified by the redirect scrubbing mode. Sequential scrubbing and scrubber redirection can be enabled independently or together.

ECC errors detected by the scrubber are logged in the MCA registers (see "Machine Check Architecture Registers" on page 146).

### 3.6.7.1 DRAM Scrub Address Low Register

**DRAM Scrub Address Low Register** **Function 3: Offset 5Ch**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–6 | ScrubAddrLo | DRAM Scrub Address Bits 31–6 | R/W | X |
| 5–1 | reserved | | R | 0 |
| 0 | ScrubReDirEn | DRAM Scrubber Redirect Enable | R/W | 0 |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

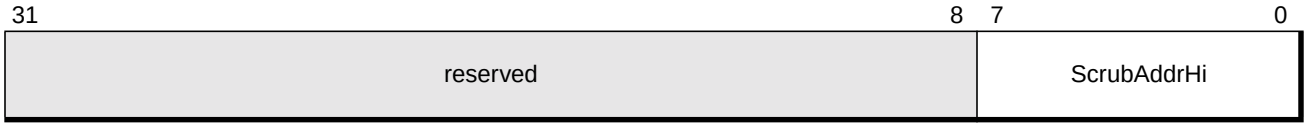
### Field Descriptions

**DRAM Scrubber Redirect Enable (ScrubReDirEn)**—Bit 0. If this bit is set, the scrubber is redirected to correct errors found during normal operation.

**DRAM Scrub Address Bits 31–6 (ScrubAddrLo)**—Bits 31–6. This field specifies the low address bits 31–6 of the next 64-byte block to be scrubbed.

### 3.6.7.2 DRAM Scrub Address High Register

**DRAM Scrub Address High Register** **Function 3: Offset 60h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | R | 0 |
| 7–0 | ScrubAddrHi | DRAM Scrub Address Bits 39–32 | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**DRAM Scrub Address Bits 39–32 (ScrubAddrHi)**—Bits 7–0. This field specifies the high address bits 39–32 of the next 64-byte block to be scrubbed.

## 3.6.8 XBAR Flow Control Buffers

The Northbridge interfaces with the CPU core, DRAM controller, and, through three HyperTransport links, to external chips.

The major Northbridge blocks are: System Request Interface (SRI), Memory Controller (MCT), and Cross Bar (XBAR). SRI interfaces with the CPU core and connects coherent HyperTransport links and noncoherent HyperTransport links. MCT maintains cache coherency and interfaces with the DRAM. XBAR is a five port switch which routes the command packets between SRI, MCT, and the three HyperTransport links. Not all HyperTransport links have to be active.

The number of buffers available for each link at the XBAR input is shown in Table 23.

**Table 23. XBAR Input Buffers**

| Link | Number of Command Buffers | Number of Data Buffers |
|---|---|---|
| HyperTransport™ Link 0-2 | 16 x 3 | 8 x 3 |
| SRI | 10 | 5 |
| MCT | 12 | 8 |
| **Total** | 70 | 37 |

XBAR command and data buffers are independent. There are 70 command buffers available, but a maximum of 64 can be used at any given time. Number of used data buffers is not restricted. The default allocation of command buffers when all HyperTransport links are present is shown in Table 24.

**Table 24. Default XBAR Command Buffer Allocation**

| Link | Number of Command Buffers |
|---|---|
| HyperTransport™ Link 0-2 | 15 x 3 |
| SRI | 8 |
| MCT | 11 |
| **Total** | 64 |

The HyperTransport™ I/O Link Specification defines four virtual channels: requests, posted requests, responses, and probes. The default virtual channel command buffer allocation is shown in Table 25. At least one command buffer should be allocated to each used virtual channel to avoid deadlock. Command buffers do not need to be allocated for a virtual channel that is not used by a link. For example, MCT does not initiate any requests, so there is no need to allocate request or posted request command buffers for an MCT link. A link should not send transactions through a virtual channel that does not have at least one command buffer allocated.

Default allocation of SRI buffers can be found in "SRI-to-XBAR Buffer Count Register" on page 116 and "Free List Buffer Count Register" on page 117. In the SRI-to-Xbar Buffer Count Register (Function 3, Offset 70h), the virtual channels are subdivided into upstream (coherent HyperTransport) and downstream (noncoherent HyperTransport) directions, since SRI must send packets into both coherent HyperTransport and noncoherent HyperTransport links. Some buffers are allocated to a free list pool to use the available resources more efficiently. The buffers in the free list pool are shared by packets in multiple virtual channels. By default, no more than one buffer is allocated to a virtual channel and the rest are allocated to the free list pool in the Free List Buffer Count Register (Function 3, Offset 7Ch). Any buffer increase to SRI can be added to the free list or allocated directly to a virtual channel. The total count allocated through the Free List Buffer Count Register and the SRI-to-Xbar Buffer Count Register cannot exceed 10.

**Table 25. Default Virtual Channel Command Buffer Allocation**

| Link | Request | Posted Request | Response | Probe | Number of Command Buffers |
|---|---|---|---|---|---|
| Coherent HyperTransport™ Links | 3 | 1 | 6 | 5 | 15 |
| Non-coherent HyperTransport™ Links | 6 | 5 | 4 | 0 | 15 |
| SRI | 2 | 3 | 3 | 0 | 8 |
| MCT | 0 | 0 | 8 | 3 | 11 |

Software can reallocate buffers by modifying buffer count registers (Function 0, Offsets 90h, B0h, D0h, Function 3, Offsets 70h, 78h, 7Ch). The new buffer counts take effect after a warm reset. Since hardware attempts to choose optimal settings, in general, these registers should not be changed.

The following information should be used for buffer reallocation:

1. Number of allocated command buffers for each link should not be greater than the number of available command buffers for that link (see Table 23). Sum of allocated command buffers for three HyperTransport links, SRI, and MCT should not be greater than 64.

2. If one HyperTransport link is not present, then other links can use all available buffers for each of them (MCT and HyperTransport links can use one additional buffer, and SRI can use two additional buffers).

3. If all links are present, some buffers from a noncoherent HyperTransport link can be reallocated to coherent HyperTransport links. Table 26 shows command buffer allocation after two response buffers from noncoherent HyperTransport link 0 are reallocated to coherent HyperTransport links 1 and 2.

**Table 26. An Example of a Non Default Virtual Channel Command Buffer Allocation**

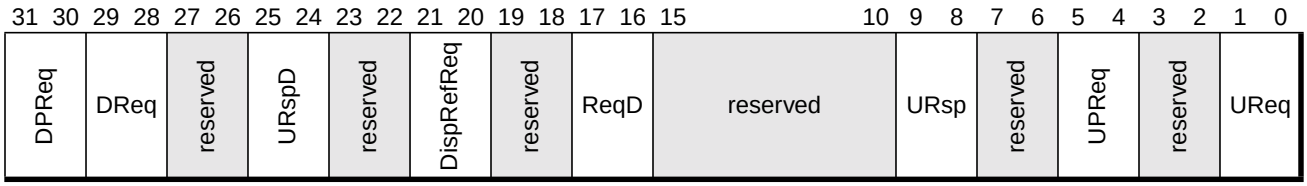
| Link | Request | Posted Request | Response | Probe | Number of Command Buffers |
|---|---|---|---|---|---|
| HyperTransport™ Link 0 | 6 | 5 | 2 | 0 | 13 |
| HyperTransport™ Link 1 | 3 | 1 | 7 | 5 | 16 |
| HyperTransport™ Link 2 | 3 | 1 | 7 | 5 | 16 |
| SRI | 2 | 3 | 3 | 0 | 8 |
| MCT | 0 | 0 | 8 | 3 | 11 |

4. If there are no noncoherent HyperTransport links, one ore more response buffers can be reallocated from MCT to HyperTransport links.

5. In a multiprocessor system, number of coherent HyperTransport response buffers should be increased first. The default buffer allocation is sufficient for a uniprocessor system.

### 3.6.8.1 SRI-to-XBAR Buffer Count Register

**SRI-to-XBAR Buffer Count Register** **Function 3: Offset 70h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–30 | DPReq | Downstream Posted Request Buffer Count | R/W | 01b |
| 29–28 | DReq | Downstream Request Buffer Count | R/W | 01b |
| 27–26 | reserved | | R | 0 |
| 25–24 | URspD | Upstream Response Data Buffer Count | R/W | 01b |
| 23–22 | reserved | | R | 0 |
| 21:20 | DispRefReq | Display Refresh Request Buffer Count | R/W | 0 |
| 19–18 | reserved | | R | 0 |
| 17–16 | ReqD | Request Data Buffer Count | R/W | 10b |
| 15–10 | reserved | | R | 0 |
| 9–8 | URsp | Upstream Response Buffer Count | R/W | 01b |
| 7–6 | reserved | | R | 0 |
| 5–4 | UPReq | Upstream Posted Request Buffer Count | R/W | 01b |
| 3–2 | reserved | | R | 0 |
| 1–0 | UReq | Upstream Request Buffer Count | R/W | 01b |

## Field Descriptions

**Upstream Request Buffer Count (UReq)**—Bits 1–0. This field defines the number of upstream request buffers available in the XBAR for use by the SRI.

**Upstream Posted Request Buffer Count (UPReq)**—Bits 5–4. This field defines the number of upstream posted request buffers available in the XBAR for use by the SRI.

**Upstream Response Buffer Count (URsp)**—Bits 9–8. This field defines the number of upstream response buffers available in the XBAR for use by the SRI.

**Request Data Buffer Count (ReqD)**—Bits 17–16. This field defines the number of request data buffers available in the XBAR for use by the SRI.

**Display Refresh Request Buffer Count (DispRefReq)**—Bits 21-20. This field defines the number of display refresh request buffers available in the XBAR for use by the SRI. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

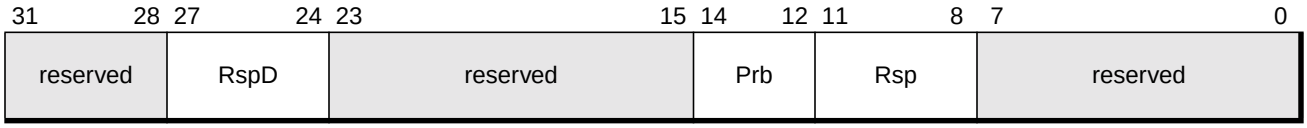
**Upstream Response Data Buffer Count (URspD)**—Bits 25–24. This field defines the number of upstream response data buffers available in the XBAR for use by the SRI.

**Downstream Request Buffer Count (DReq)**—Bits 29–28. This field defines the number of downstream request buffers available in the XBAR for use by the SRI.

**Downstream Posted Request Buffer Count (DPReq)**—Bits 31–30. This field defines the number of downstream posted request buffers available in the XBAR for use by the SRI.

### 3.6.8.2 MCT-to-XBAR Buffer Count Register

**MCT-to-XBAR Buffer Count Register** **Function 3: Offset 78h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–28 | reserved | | R | 0 |
| 27–24 | RspD | Response Data Buffer Count | R/W | 8h |
| 23–15 | reserved | | R | 0 |
| 14–12 | Prb | Probe Buffer Count | R/W | 011b |
| 11–8 | Rsp | Response Buffer Count | R/W | 8h |
| 7–0 | reserved | | R | 0 |

#### Field Descriptions

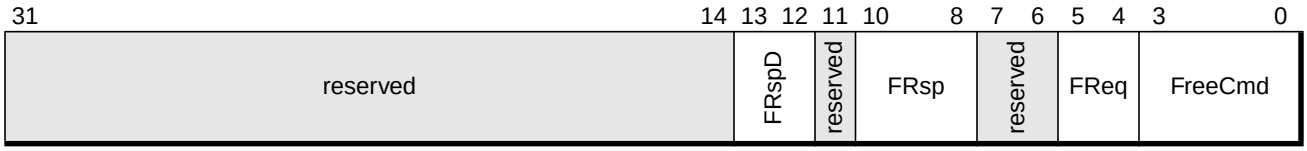
**Response Buffer Count (Rsp)**—Bits 11–8. This field defines the number of response buffers available in the XBAR for use by the MCT.

**Probe Buffer Count (Prb)**—Bits 14–12. This field defines the number of probe buffers available in the XBAR for use by the MCT.

**Response Data Buffer Count (RspD)**—Bits 27–24. This field defines the number of response data buffers available in the XBAR for use by the MCT.

### 3.6.8.3 Free List Buffer Count Register

**Free List Buffer Count Register** **Function 3: Offset 7Ch**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–14 | reserved | | R | 0 |
| 13–12 | FRspD | SRI to XBAR Free Response Data Buffer Count | R/W | 10b |
| 11 | reserved | | R | 0 |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 10–8 | FRsp | SRI to XBAR Free Response Buffer Count | R/W | 010b |
| 7–6 | reserved | | R | 0 |
| 5–4 | FReq | SRI to XBAR Free Request Buffer Count | R/W | 01b |
| 3–0 | FreeCmd | SRI Free Command Buffer Count | R/W | Bh |

### Field Descriptions

**SRI Free Command Buffer Count (FreeCmd)**—Bits 3–0. This field defines the number of free list request or posted request buffers available in the SRI for use by the XBAR or CPU.

**SRI to XBAR Free Request Buffer Count (FReq)**—Bits 5–4. This field defines the number of free list request buffers available in the XBAR for use by the SRI.

**SRI to XBAR Free Response Buffer Count (FRsp)**—Bits 10–8. This field defines the number of free list response buffers available in the XBAR for use by the SRI.

**SRI to XBAR Free Response Data Buffer Count (FRspD)**—Bits 13–12. This field defines the number of free list response data buffers available in the XBAR for use by the SRI.

## 3.6.9 XBAR-to-SRI Buffer Count Register

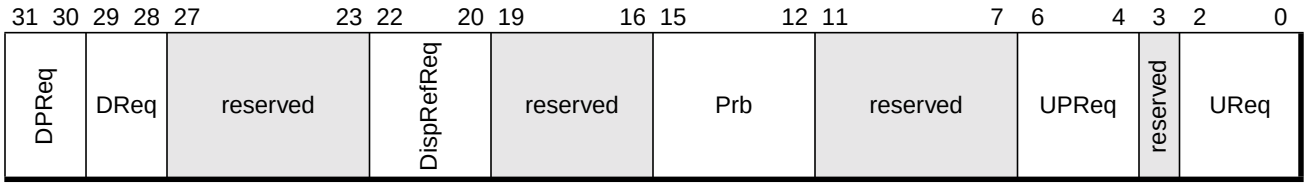
The Cross Bar Switch to System Request Interface (XBAR-to-SRI) Buffer Count register specifies the number of command buffers for each virtual channel available in the SRI for use by the XBAR. These buffers store commands for traffic routed to the SRI by the XBAR. The buffer counts take effect on the next warm reset.

Since the XBAR must route packets in both the coherent HyperTransport technology fabric and down the noncoherent HyperTransport chains, the usual virtual channels are further subdivided into upstream (from noncoherent HyperTransport) and downstream (to noncoherent HyperTransport) directions. In order to make more efficient use of the available resources, some of the buffers are allocated onto a free list pool in which case these buffers can be used by packets issued in either direction. See "Free List Buffer Count Register" on page 117. Any increase in the number of fixed allocated buffers in the XBAR-to-SRI Buffer Count register must be compensated by a corresponding reduction of the number of free list buffers specified in the Free List Buffer Count register.

When modifying buffer counts care must be taken to allocate a minimum number of buffers to each virtual channel to avoid deadlock. Requests and Posted requests in both directions require at least one buffer and Probes require at least two buffers to avoid deadlock. Since hardware attempts to choose optimal settings, this register should not in general need to be changed.

***Note:*** *When the Probe Buffer Count is changed, care must be taken to avoid generation of system management events from the time the new value is written into this register until it takes effect (on the next warm reset).*

### XBAR-to-SRI Buffer Count Register — Function 3: Offset 74h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–30 | DPReq | Downstream Posted Request Buffer Count | R/W | 01b |
| 29–28 | DReq | Downstream Request Buffer Count | R/W | 01b |
| 27–23 | reserved | | R | 0 |
| 22:20 | DispRefReq | Display Refresh Request Buffer Count | R/W | 0 |
| 19–16 | reserved | | R | 0 |
| 15–12 | Prb | Probe Buffer Count | R/W | 8h |
| 11–7 | reserved | | R | 0 |
| 6–4 | UPReq | Upstream Posted Request Buffer Count | R/W | 001b |
| 3 | reserved | | R | 0 |
| 2–0 | UReq | Upstream Request Buffer Count | R/W | 001b |

#### Field Descriptions

**Upstream Request Buffer Count (UReq)**—Bits 2–0. This field defines the number of upstream request buffers available in the SRI for use by the XBAR.

**Upstream Posted Request Buffer Count (UPReq)**—Bits 6–4. This field defines the number of upstream posted request buffers available in the SRI for use by the XBAR.

**Probe Buffer Count (Prb)**—Bits 15–12. This field defines the number of probe buffers available in the SRI for use by the XBAR.

**Display Refresh Request Buffer Count (DispRefReq)**—Bits 22-20. This field defines the number of display refresh request buffers available in the SRI for use by the XBAR. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**Downstream Request Buffer Count (DReq)**—Bits 29–28. This field defines the number of downstream request buffers available in the SRI for use by the XBAR.

**Downstream Posted Request Buffer Count (DPReq)**—Bits 31–30. This field defines the number of downstream posted request buffers available in the SRI for use by the XBAR.

## 3.6.10 Display Refresh Flow Control Buffers

The Northbridge has a separate logical path from HyperTransport links to the system memory for display refresh requests. This is necessary to support the bandwidth and latency requirements of

display refresh requests in systems where the display-refresh frame buffer is located in the system memory.

A HyperTransport packet is defined as a display-refresh request packet when the read request has isochronous bit, PassPW bit, and RspPassPW bit set, and coherent bit and SeqID cleared. All display-refresh requests are marked as high priority requests and will have higher priority than other requests. In order to accomplish a dedicated logical path to system memory a minimum number of buffers have to be allocated in various queues.

DispRefReq (Function 3, Offset 70h) and DispRefReq (Function 3, Offset 74h) need to be set to allocate the buffers for display refresh requests. At least one buffer should be allocated for display refresh requests to avoid deadlock. Usually more than one buffer may be required to support the necessary bandwidth. The buffer count written to the registers takes effect after a warm reset. See "SRI-to-XBAR Buffer Count Register" on page 116 and "XBAR-to-SRI Buffer Count Register" on page 118 for more information on setting these registers.

## 3.6.11 Power Management Control Registers

These registers specify the processor response to STPCLK and HALT power management events. Each STPCLK request received contains a 3-bit System Management Action Field (SMAF) which is used as an index into the Power Management Control Low/High registers to select a Power Management Mode (PMM) value to use for the duration of the STPCLK event. A SMAF value of 000 selects PMM0, a SMAF value of 001 selects PMM1, and so on. HALT is hardwired to use PMM7.

### 3.6.11.1 Power Management Mode (PMM) Value

The PMM value contains the following fields:

| Bit | Name | Function | R/W |
|---|---|---|---|
| 6–4 | ClkSel | Clock Divisor Select | R/W |
| 3 | reserved | | R |
| 2 | FidVidEn | FID/VID Change Enable | R/W |
| 1 | NBLowPwrEn | Northbridge Low Power Enable | R/W |
| 0 | CPULowPwrEn | CPU Low Power Enable | R/W |

#### PMM Value Field Descriptions

**Clock Divisor Select (ClkSel)**—Bits 6–4. This field specifies the divisor to use when ramping down the CPU clock or the Northbridge clock.

| | |
|---|---|
| 000b = Divide by 8 | 100b = Divide by 128 |
| 001b = Divide by 16 | 101b = Divide by 256 |
| 010b = Divide by 32 | 110b = Divide by 512 |
| 011b = Divide by 64 | 111b = reserved |

**FID/VID Change Enable (FidVidEn)**—Bit 2. Enables a change in Frequency ID (FID) and/or Voltage ID (VID). The CPU and the Northbridge clocks must also be ramped down (see NBLowPwrEn and CPULowPwrEn).

0 = FID/VID change disabled
1 = FID/VID change enabled

**Northbridge Low Power Enable (NBLowPwrEn)**—Bit 1. Causes the Northbridge clock to ramp down according to the clock divisor specified in ClkSel and puts the DRAM in self-refresh mode. The CPU clock must also be ramped down (see CPULowPwrEn)

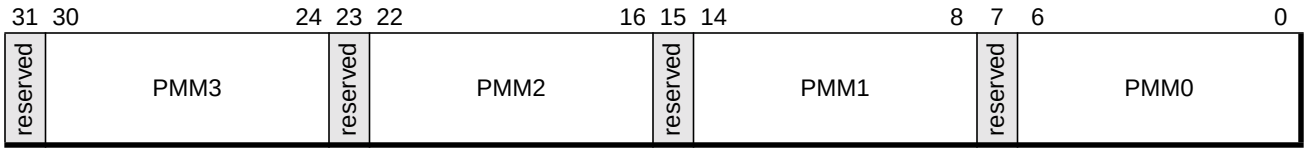
0 = Northbridge low power disabled
1 = Northbridge low power enabled

**CPU Low Power Enable (CPULowPwrEn)**—Bit 0. Causes the CPU clock to ramp down according to the clock divisor specified in ClkSel.

0 = CPU low power disabled
1 = CPU low power enabled

### 3.6.11.2 Power Management Control Low Register

**Power Management Control Low Register** **Function 3: Offset 80h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31 | reserved | | R | 0 |
| 30–24 | PMM3 | Power Management Mode 3 | R/W | 0 |
| 23 | reserved | | R | 0 |
| 22–16 | PMM2 | Power Management Mode 2 | R/W | 0 |
| 15 | reserved | | R | 0 |
| 14–8 | PMM1 | Power Management Mode 1 | R/W | 0 |
| 7 | reserved | | R | 0 |
| 6–0 | PMM0 | Power Management Mode 0 | R/W | 0 |

#### Field Descriptions

**Power Management Mode 0 (PMM0)**—Bits 6–0. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

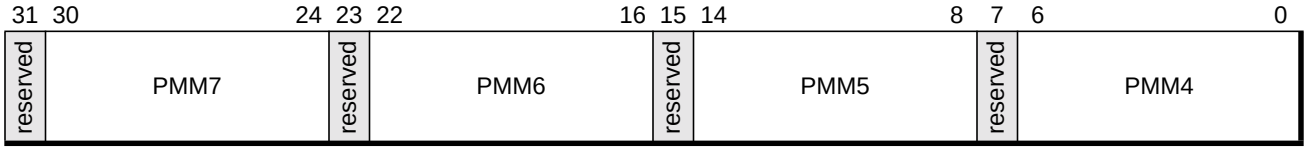
**Power Management Mode 1 (PMM1)**—Bits 14–8. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

**Power Management Mode 2 (PMM2)**—Bits 22–16. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

**Power Management Mode 3 (PMM3)**—Bits 30–24. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

### 3.6.11.3 Power Management Control High Register

**Power Management Control High Register** **Function 3: Offset 84h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31 | reserved | | R | 0 |
| 30–24 | PMM7 | Power Management Mode 7 | R/W | 0 |
| 23 | reserved | | R | 0 |
| 22–16 | PMM6 | Power Management Mode 6 | R/W | 0 |
| 15 | reserved | | R | 0 |
| 14–8 | PMM5 | Power Management Mode 5 | R/W | 0 |
| 7 | reserved | | R | 0 |
| 6–0 | PMM4 | Power Management Mode 4 | R/W | 0 |

#### Field Descriptions

**Power Management Mode 4 (PMM4)**—Bits 6–0. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

**Power Management Mode 5 (PMM5)**—Bits 14–8. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

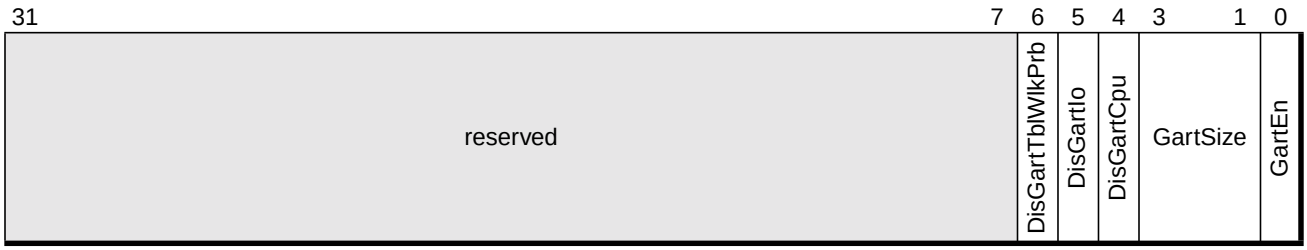
**Power Management Mode 6 (PMM6)**—Bits 22–16. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

**Power Management Mode 7 (PMM7)**—Bits 30–24. See "Power Management Mode (PMM) Value" on page 120 for a description of these bits.

## 3.6.12 GART Aperture Control Register

This register contains the aperture size and enable for the graphics aperture relocation table (GART) mechanism.

## GART Aperture Control Register — Function 3: Offset 90h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–7 | reserved | | R | 0 |
| 6 | DisGartTblWlkPrb | Disable GART Table Walk Probes | R/W | 0 |
| 5 | DisGartIo | Disable GART I/O Accesses | R/W | 0 |
| 4 | DisGartCpu | Disable GART CPU Accesses | R/W | 0 |
| 3–1 | GartSize | GART Size | R/W | 0 |
| 0 | GartEn | GART Enable | R/W | 0 |

### Field Descriptions

**GART Enable (GartEn)**—Bit 0. Enables GART address translation for accesses falling within the GART aperture. GartAperBaseAddr (Function 3, Offset 94h) and other related registers should be initialized before GartEn is set.

**GART Size (GartSize)**—Bits 3–1. Defines the virtual address space to be allocated to the GART.

000b = 32 Mbytes
001b = 64 Mbytes
010b = 128 Mbytes
011b = 256 Mbytes
100b = 512 Mbytes
101b = 1 Gbyte
110b = 2 Gbyte
111b = reserved

**Disable GART CPU Accesses (DisGartCpu)**—Bit 4. Disables requests from CPUs from accessing the GART.

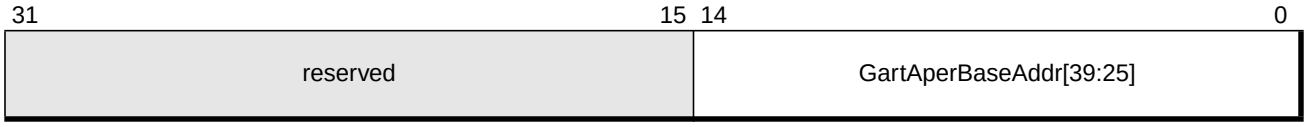
**Disable GART I/O Accesses (DisGartIo)**—Bit 5. Disables requests from I/O devices from accessing the GART.

**Disable GART Table Walk Probes (DisGartTblWlkPrb)**—Bit 6. Disables generation of probes for GART table walks. This bit may be set to improve performance in cases where the GART table entries are in address space which is marked uncacheable in processor MTRRs or page tables. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

## 3.6.13 GART Aperture Base Register

This register contains the base address of the aperture for the graphics aperture relocation table (GART). The GART aperture base along with the GART aperture size define the GART aperture address window. BIOS can place the GART aperture below the 4-gigabyte level in address space in order to support legacy operating systems and legacy AGP cards (that do not support 64-bit address space).

**GART Aperture Base Register** **Function 3: Offset 94h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–15 | reserved | | R | 0 |
| 14–0 | GartAperBaseAddr[39:25] | GART Aperture Base Address Bits 39–25 | R/W | X |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

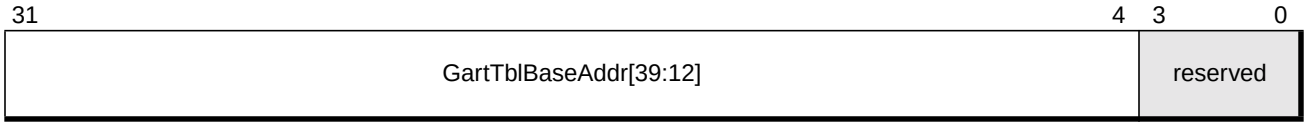
**GART Aperture Base Address Bits 39–25 (GartAperBaseAddr[39:25])**—Bits 14–0. These bits are used to compare to the incoming addresses to determine if they are in the aperture range. They are active based on the aperture size. The remaining address bits are assumed to be 0.

[39:25] = 32 Mbytes
[39:26] = 64 Mbytes
[39:27] = 128 Mbytes
[39:28] = 256 Mbytes
[39:29] = 512 Mbytes
[39:30] = 1 Gbytes
[39:31] = 2 Gbytes

## 3.6.14 GART Table Base Register

This register contains the base address of the translation table for the graphics aperture relocation table (GART). The GART table base points to the base address of a table of 32-bit GART page table entries (PTEs) that contain the physical addresses to use for an incoming address that falls within the GART aperture.

**GART Table Base Register** **Function 3: Offset 98h**



| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–4 | GartTblBaseAddr[39:12] | GART Table Base Address Bits 39–12 | R/W | X |
| 3–0 | reserved | | R | 0 |

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**GART Table Base Address Bits 39–12 (GartTblBaseAddr[39:12])**—Bits 31–4. These bits point to the base of the table of GART PTEs to be used for GART address translation. Table 27 shows the organization of each 32-bit PTE in the GART table.

**Table 27. GART PTE Organization**

| Bits | Description |
|---|---|
| 0 | Valid |
| 1 | Coherent |
| 3–2 | reserved |
| 11–4 | PhysAddr[39:32] |
| 31–12 | PhysAddr[31:12] |

## 3.6.15 GART Cache Control Register

This register controls the GART cache, which is used to cache the most recently translated GART aperture accesses. The GART cache is enabled automatically whenever the GART aperture is enabled.

**GART Cache Control Register** **Function 3: Offset 9Ch**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–2 | reserved | | R | 0 |
| 1 | GartPteErr | GART PTE Error | R/WC | 0 |
| 0 | InvGart | Invalidate GART | R/W | 0 |

### Field Descriptions

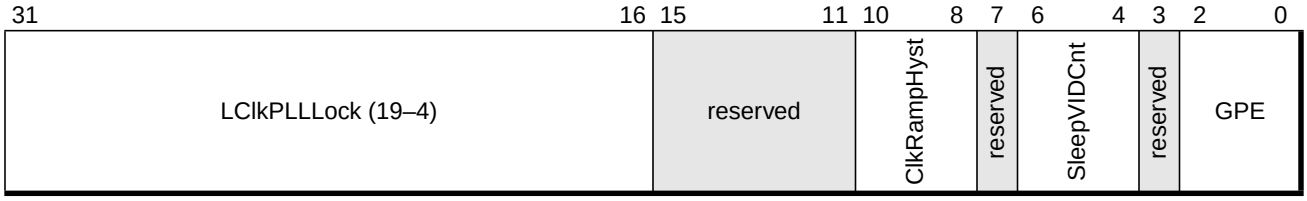
**Invalidate GART (InvGart)**—Bit 0. Setting this bit causes the GART cache to be invalidated. This bit is cleared by hardware when the invalidation is complete.

**GART PTE Error (GartPteErr)**—Bit 1. This bit is set when an invalid PTE is encountered during a table walk. Cleared by writing a 1.

## 3.6.16 Clock Power/Timing Low Register

This register controls the transition times between various voltage and frequency states. The value of this register is maintained through a warm reset and is initialized to 0 on a cold reset.

**Clock Power/Timing Low Register** **Function 3: Offset D4h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | LClkPLLLock | HyperTransport CLK PLL Lock Counter Bits 19–4 | R/W | 0 |
| 15–11 | reserved | | R | 0 |
| 10-8 | ClkRampHyst | Clock Ramp Hysteresis | R/W | 0 |
| 7 | reserved | | R | 0 |
| 6–4 | SleepVIDCnt | Time to Drive Sleep VID | R/W | 0 |
| 3 | reserved | | R | 0 |
| 2–0 | GPE | Good Phase Error | R/W | 0 |

### Field Descriptions

**Good Phase Error (GPE)**—Bits 2–0. This field defines the BCLK PLL time until good phase error. Counting occurs when at full frequency.

Revision B and earlier revision encodings are:

000b = 16 system clocks
001b = 400 system clocks
010b = 800 system clocks
011b = 1200 system clocks
100b = 1600 system clocks
101b = 2000 system clocks
110b = 2400 system clocks
111b = 4095 system clocks

Revision C encodings are:

000b = reserved
001b = 200 system clocks (1us)
010b = 400 system clocks (2us)
011b = 600 system clocks (3us)

100b = 800 system clocks (4us)
101b = 1600 system clocks (8us)
110b = 3000 system clocks (15us)
111b = 20000 system clocks (100us)

**Time to Drive Sleep VID (SleepVIDCnt)**—Bits 6–4. This field defines the delay time to Drive Sleep VID. This field indicates when all CPUs have ramped down GCLK/BCLK grids before swapping in the new VID. In a uniprocessor system or a multiprocessor system where each process controls its own voltage regulator, this field can be programmed to 000b. This counter clocks PLL system clocks (5ns).

000b = 0
001b = 200 system clocks
010b = 400 system clocks
011b = 600 system clocks
100b = 800 system clocks
101b = 10000 system clocks
110b = 100000 system clocks
111b = 200000 system clocks

**Clock Ramp Hysteresis (ClkRampHyst)**—Bits 10-8. A non-zero value in this field enables a hysteresis time which prevents the CPU clock grid from being ramped down after processing a probe. It avoids unnecessary changes of the CPU clock grid when the probe arrival rate is relatively low. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

000b = 0
001b = 125 ns
010b = 250 ns
011b = 375 ns
100b = 500 ns
101b = 750 ns
110b = 1000 ns
111b = 2000 ns

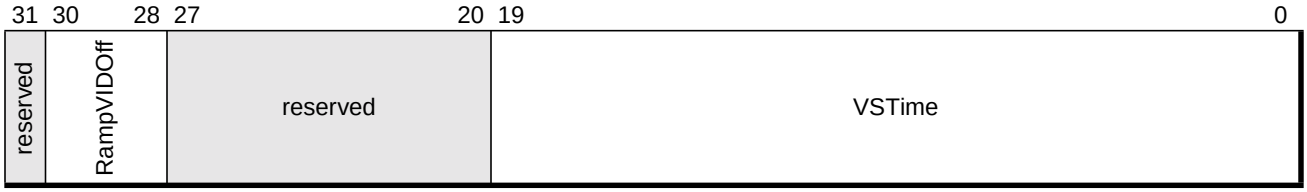
**HyperTransport CLK PLL Lock Counter Bits 19–4 (LClkPLLLock)**—Bits 31–16. This bit field indicates how long it takes for the slowest HyperTransport technology clock PLL to ramp to its new frequency and lock.

The 16 bits represent the most significant 16 bits of a 20-bit value. The number reflects the number of system bus clocks (5 ns) to wait. It is only necessary to wait a short amount of time in the event the frequency change is one that maintains the VCO frequency. In this case, the PLL will be re-locked quickly.

## 3.6.17 Clock Power/Timing High Register

This register controls the transition times between various voltage and frequency states. The value of this register is maintained through a warm reset and is initialized to 0 on a cold reset.

**Clock Power/Timing High Register** **Function 3: Offset D8h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31 | reserved | | R | 0 |
| 30–28 | RampVIDOff | Ramp VID Offset | R/W | 0 |
| 27–20 | reserved | | R | 0 |
| 19–0 | VSTime | Voltage Regulator Stabilization Time | R/W | 0 |

### Field Descriptions

**Voltage Regulator Stabilization Time (VSTime)**—Bits 19–0. This field indicates when the voltage regulator is stable at the Ramp VID before ramping up the clocks. The count is the number of 5-ns system clock cycles.

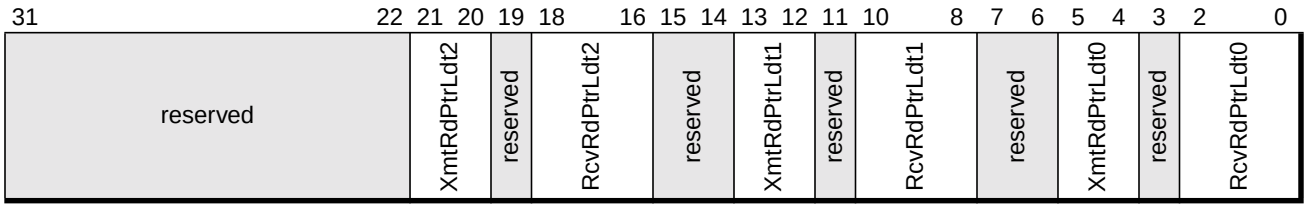
**Ramp VID Offset (RampVIDOff)**—Bits 30–28. Defines the amount of extra voltage required while the PLL is ramping for low power states. This "over-voltage" is applied only until the PLL has phase locked to within specifications.

000b = 0 mV
001b = 25 mV
010b = 50 mV
011b = 75 mV
100b = 100 mV
101b = 125 mV
110b = 150 mV
111b = 175 mV

## 3.6.18 HyperTransport™ FIFO Read Pointer Optimization Register

This register allows the separation of read/write pointers in the HyperTransport technology receive/transmit FIFOs to be changed from their default settings. The value of this register is maintained through a warm reset and is initialized to 0 on a cold reset.

## HyperTransport™ FIFO Read Pointer Optimization Register Function 3: Offset DCh




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–22 | reserved | | R | 0 |
| 21–20 | XmtRdPtrLdt2 | Change Read Pointer For HyperTransport Link 2 Transmitter | R/W | 0 |
| 19 | reserved | | R | 0 |
| 18–16 | RcvRdPtrLdt2 | Change Read Pointer For HyperTransport Link 2 Receiver | R/W | 0 |
| 15–14 | reserved | | R | 0 |
| 13–12 | XmtRdPtrLdt1 | Change Read Pointer For HyperTransport Link 1 Transmitter | R/W | 0 |
| 11 | reserved | | R | 0 |
| 10–8 | RcvRdPtrLdt1 | Change Read Pointer For HyperTransport Link 1 Receiver | R/W | 0 |
| 7–6 | reserved | | R | 0 |
| 5–4 | XmtRdPtrLdt0 | Change Read Pointer For HyperTransport Link 0 Transmitter | R/W | 0 |
| 3 | reserved | | R | 0 |
| 2–0 | RcvRdPtrLdt0 | Change Read Pointer For HyperTransport Link 0 Receiver | R/W | 0 |

### Field Descriptions

**Change Read Pointer For HyperTransport Link 0 Receiver (RcvRdPtrLdt0)**—Bits 2–0. See RcvRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 0 Transmitter (XmtRdPtrLdt0)**—Bits 5–4. See XmtRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 1 Receiver (RcvRdPtrLdt1)**—Bits 10–8. See RcvRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 1 Transmitter (XmtRdPtrLdt1)**—Bits 13–12. See XmtRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 2 Receiver (RcvRdPtrLdt2)**—Bits 18–16. Moves the read pointer for the HyperTransport receive FIFO closer to the write pointer thereby reducing latency through the receiver.

000b = RdPtr assigned by hardware
001b = Move RdPtr closer to WrPtr by 1 HyperTransport clock period

010b = Move RdPtr closer to WrPtr by 2 HyperTransport clock periods
011b = Move RdPtr closer to WrPtr by 3 HyperTransport clock periods
100b = Move RdPtr closer to WrPtr by 4 HyperTransport clock periods
101b = Move RdPtr closer to WrPtr by 5 HyperTransport clock periods
110b = Move RdPtr closer to WrPtr by 6 HyperTransport clock periods
111b = Move RdPtr closer to WrPtr by 7 HyperTransport clock periods

AMD recommends setting this field to 5 for all coherent HyperTransport links and noncoherent HyperTransport links to AMD chipsets. Optimal value for noncoherent HyperTransport links to other chipsets needs to be determined by the developer and tested to ensure stability.

**Change Read Pointer For HyperTransport Link 2 Transmitter (XmtRdPtrLdt2)**—Bits 21–20. Moves the read pointer for the HyperTransport technology transmit FIFO closer to the write pointer thereby reducing latency through the transmitter.

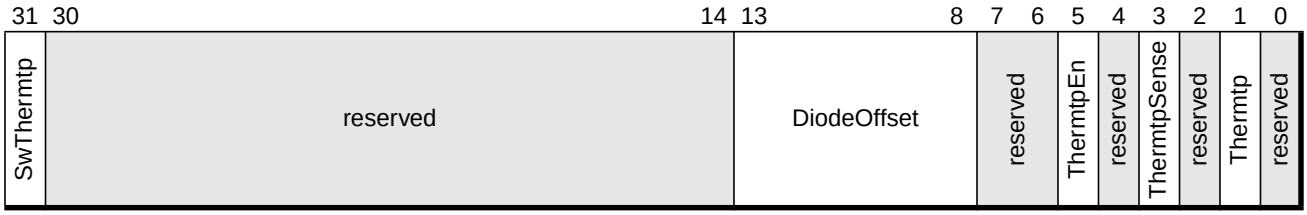
00b = RdPtr assigned by hardware
01b = Move RdPtr closer to WrPtr by 1 HyperTransport clock period
10b = Move RdPtr closer to WrPtr by 2 HyperTransport clock periods
11b = Move RdPtr closer to WrPtr by 3 HyperTransport clock periods

AMD recommends setting this field to 2 for all coherent HyperTransport links and noncoherent HyperTransport links to AMD chipsets. Optimal value for noncoherent HyperTransport links to other chipsets needs to be determined by the developer and tested to ensure stability.

## 3.6.19 Thermtrip Status Register

The Thermtrip Status register provides status information regarding the THERMTRIP thermal sensor.

**Thermtrip Status Register** **Function 3: Offset E4h**




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31 | SwThermtp | Software Thermtrip | W | 0 |
| 30–14 | reserved | | R | 0 |
| 13–8 | DiodeOffset | Diode Offset | R | |
| 7-6 | reserved | | R | 0 |
| 5 | ThermtpEn | Thermtrip Enabled | R | |
| 4 | reserved | | R | 0 |
| 3 | ThermtpSense | Thermtrip Sense | R | |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 2 | reserved | | R | 0 |
| 1 | Thermtp | Thermtrip | R | |
| 0 | reserved | | R | 0 |

### Field Descriptions

**Thermtrip (Thermtp)**—Bit 1. Set to 1 if a temperature sensor trip occurs and was enabled.

**Thermtrip Sense (ThermtpSense)**—Bit 3. Set to 1 if a temperature sensor trip occurs.

**Thermtrip Enabled (ThermtpEn)**—Bit 5. Indicates that the thermtrip temperature sensor is enabled. When this bit is set to 1, a THERMTRIP High event will cause the hardware to shut down the PLL, assert the THERMTRIP output pin and set the ThermtpHi bit. The ThermtpSense bit is set for a THERMTRIP High event, irrespective of the state of ThermtpEn.

**Diode Offset (DiodeOffset[5:0])**—Bits 13–8.Thermal diode offset is used to correct temperature measurement made by an external temperature sensor. The offset is in 1 degree Celsius increments and it should be subtracted from the temperature measurement. A correction to the offset may be needed for some temperature sensors. Contact the temperature sensor vendor to determine whether an offset correction is needed. The maximum allowable offset is provided in the appropriate processor data sheet, and the maximum offset can vary for different processors.

**Software Thermtrip (SwThermtp)**—Bit 31. Writing a 1 to this bit position induces a THERMTRIP event. This bit is write-only and returns 0 when read. This is a diagnostic bit, and it should be used for testing purposes only.

## 3.6.20 Northbridge Capabilities Register

The Northbridge Capabilities register indicates whether or not this Northbridge is capable of certain behavior.

### Northbridge Capabilities Register — Function 3: Offset E8h




| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 31–9 | reserved | | R | 0 |
| 8 | MemCntCap | Memory Controller Capable | R | |

| Bits | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 7 | reserved | | R | 0 |
| 6–5 | DramFreq | Maximum DRAM Frequency | R | |
| 4 | ChipKillEccCap | Chip-Kill ECC Capable | R | |
| 3 | EccCap | ECC Capable | R | |
| 2 | BigMPCap | Big MP Capable | R | |
| 1 | MPCap | MP Capable | R | |
| 0 | 128BitCap | 128-Bit DRAM Capable | R | |

## Field Descriptions

**128-Bit DRAM Capable (128BitCap)**—Bit 0. This bit is set to 1 if the Northbridge is capable of supporting a 128-bit DRAM interface.

**MP Capable (MPCap)**—Bit 1. This bit is set to 1 if the Northbridge is capable of supporting multiprocessor systems.

**Big MP Capable (BigMPCap)**—Bit 2. This bit is set to 1 if the Northbridge is capable of supporting multiprocessor systems greater than DP.

**ECC Capable (EccCap)**—Bit 3. This bit is set to 1 if the Northbridge is capable of supporting ECC.

**Chip-Kill ECC Capable (ChipKillEccCap)**—Bit 4. This bit is set to 1 if the Northbridge is capable of supporting chip-kill ECC.

**Maximum DRAM Frequency (DramFreq)**—Bits 6–5. Indicates the maximum DRAM frequency supported.

00b = No limit
01b = 166 MHz
10b = 133 MHz
11b = 100 MHz

**Memory Controller Capable (MemCntCap)**—Bit 8. This bit is set to 1 if the Northbridge is capable of supporting an on-chip memory controller.

# 4 DRAM Configuration

The DRAM interface controller supports the following programmable features:

- AMD Opteron™ processor 900+ package pin devices (64-bit or 128-bit interface).
- AMD Athlon™ 64 processor 700+ package pin devices (only 64-bit interface); up to three unbuffered DIMMs or up to four registered DIMMs. Refer to the processor data sheet for restrictions with unbuffered 3-DIMM support.
- 100 MHz (DDR200), 133 MHz (DDR266), 166 MHz (DDR333), and 200 MHz (DDR400) devices.
- 4-bit, 8-bit, and 16-bit DRAM devices; systems in which the DIMMs have different widths.
- DIMM sizes from 32 Mbytes (64 Mbit x 16) to 2 Gbytes (1 Gbit x 4)
- Interleaving memory within DIMMs.
- Stacked registered DIMMs and angled DIMMs.
- ECC checking.
- 32-byte and 64-byte burst lengths.
- The DRAM controller may be disabled.
- Statically configurable control over the time a page is allowed to remain open while idle.
- Optional dynamic control of page closure.
- Mode to control the on-chip DLL and programmable delay lines.

The DRAM controller does not make use of every feature that DRAM devices implement. The following list is the set of features *not* supported by the DRAM controller.

- Burst terminate command is not used (nor are bursts terminated by any other means).
- Auto precharge command is not used.
- Active or precharge power down is not supported (self-refresh mode or other processor power down states are used to reduce power).
- Data masking is supported only on Revision C processors and later.
- The two DIMMs making up the 128-bit-wide data bus must be of the same size and configuration.

# 4.1 Programming Interface

This section describes how to program various DRAM controller registers. There are two principal areas of configuration:

- Configuration state information obtained via the DIMM Serial Presence Detect (SPD) ROMs.
- All other configuration state information.

## 4.1.1 SPD ROM-Based Configuration

The SPD device is an EEPROM on the DIMM encoded by the DIMM manufacturer. The description of the EEPROM is usually provided on a data sheet for the DIMM itself along with data describing the memory devices used. The data describes configuration and speed characteristics of the DIMM and the SDRAM components mounted on the DIMM. The data sheet also contains the DIMM byte values that are encoded in the SPD on the DIMM.

BIOS acquires the values encoded in the SPD ROM through the I/O hub, which obtains the information through a secondary device connected to the I/O hub through the SMBus. This secondary device communicates with the DIMM by means of the $I^2C$ bus.

The SPD ROM provides values for several DRAM timing parameters that are required by the DRAM controller. These parameters are:

- tCL: (CAS latency)
- tRC: Active-to-Active/Auto Refresh command period
- tRFC: Auto-Refresh-to-Active/Auto Refresh command period
- tRCD: Active-to-Read-or-Write delay
- tRRD: Active-Bank-A to-Active-Bank-B delay
- tRAS: Active-to-Precharge delay
- tRP: Precharge time
- tREF: Refresh interval (function of DRAM density)

### 4.1.1.1 tCL (CAS Latency)

The number of memory clocks it takes a DRAM to return data after the read CAS_L is asserted depends on the memory clock frequency. The value that BIOS programs into the memory controller is a function of the target clock frequency. The target clock frequency is determined from the supported CAS latencies at given clock frequencies of each DIMM. A suggested algorithm is as follows:

1. Determine all CAS latencies supported by each installed DIMM, defined in SPD byte 18. One bit corresponds to each supported CAS latency. SPD byte 18 specifies CAS latencies with which devices on the DIMM can reliably operate. BIOS should not assume that a device can operate

with either slower or faster CAS latency than those specified by the SPD. Typically, all DDR DIMMs support CAS latencies of 2 and 2.5. Some DDR DIMMs may support CAS latencies of 3. The DRAM controller is designed to support these CAS latencies. The SPD ROM allows the manufacturer to indicate support for CAS latency 3.5. This value is not supported by the DRAM controller and should be discarded.

2. Determine the maximum clock frequency at each supported CAS latency for each DIMM. The minimum cycle time for the highest, the second highest, and the third highest supported CAS latencies is defined in SPD byte 9, 23, and 25, respectively. There is a possibility of incorrectly programmed SPDs such that a cycle time from SPD byte 23 or 25, corresponding to a set bit in SPD byte 18, is unimplemented. BIOS should discard this pair. The minimum cycle time has 1/10ns granularity.
3. Determine the best CAS latency and clock frequency combination. Find the highest clock frequency supported by the slowest DIMM and determine the CAS latency at that operating frequency. It is necessary to choose the highest CAS latency supported by all the DIMMs at the target frequency.

The processor also provides silicon fuses that limit the frequency of the DRAM clock. BIOS can read this information from the Northbridge Capabilities register (function 3, offset E8h[6:5]). BIOS needs to assume that the highest supported frequency is given by the slowest DIMM or the Maximum DRAM Frequency fuses, whichever is lower.

There is a possibility that there are two options: a 166 MHz tCL = 3 configuration or a 133 MHz tCL = 2 configuration. The above algorithm would result in selecting the 166 MHz configuration, but it is known through performance analysis that this is not the best choice. Whenever tCL for the higher frequency is 1 greater than tCL for the next lowest frequency, BIOS should select the lower frequency. If the tCL difference is 0.5, then the higher frequency should be selected.

#### 4.1.1.2 tRCD (RAS-to-CAS Delay)

This parameter is defined in SPD byte 29 and it has 1/4ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in "tCL (CAS Latency)" on page 134. Typically, this value is 48h for DDR333, which maps to 18 ns or 3 clocks.

#### 4.1.1.3 tRAS (Active-to-Precharge Delay)

This parameter is defined in SPD byte 30 and it has 1-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in "tCL (CAS Latency)" on page 134. Typically, this value is 2Ah for DDR333, which maps to 42 ns or 7 clocks.

#### 4.1.1.4 tRP (Precharge Command Period)

This parameter is defined in SPD byte 27 and it has 1/4-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in "tCL (CAS Latency)" on page 134. Typically, this value is 48h for DDR333, which maps to 18 ns or 3 clocks.

#### 4.1.1.5 tRC (Active-to-Active/Auto-Refresh Command Period)

This parameter is defined in SPD byte 41 and it has 1-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in "tCL (CAS Latency)" on page 134. Typically, this value is 3Ch for DDR333, which maps to 60 ns or 10 clocks.

Some DIMMs may not include the tRC value in byte 41. In this case, BIOS will read either 00h or FFh from the ROM and it should do the following:

- At 100 MHz, tRC is assumed to be 46h, which maps to 70 ns or 7 clocks.
- At 133 MHz, tRC is assumed to be 41h, which maps to 65 ns or 9 clocks.
- At 166 MHZ, tRC is assumed to be 3Ch, which maps to 60 ns or 10 clocks.
- At 200 MHZ, tRC is assumed to be 37h, which maps to 55 ns or 11 clocks.

#### 4.1.1.6 tRRD (Active-to-Active of a Different Bank)

This parameter is defined in SPD byte 28 and it has 1/4-ns granularity. BIOS should read this value and convert into a number of DRAM clocks using the target frequency obtained in "tCL (CAS Latency)" on page 134. Typically, this value is 30h for DDR333, which maps to 12 ns or 2 clocks.

#### 4.1.1.7 tRFC (Auto-Refresh-to-Active/Auto-Refresh Command Period)

This parameter is defined in SPD byte 42 and it has 1-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in "tCL (CAS Latency)" on page 134. Typically, this value is 48h for DDR333 DRAMs between 64-Mbit and 512-Mbit and 78h for DDR333 1-Gbit DRAMS.

Some DIMMs may not include the tRFC value in byte 42. In this case, BIOS will read either 00h or FFh from the ROM and it should do the following:

- At 100 MHz, tRFC is assumed to be 50h, which maps to 80 ns or 8 clocks.
- At 133 MHz, tRFC is assumed to be 4Bh, which maps to 75 ns or 10 clocks.
- At 166 MHz, tRFC is assumed to be 48h, which maps to 72 ns or 12 clocks for 512-Mbit devices or smaller. For 1-Gbit devices, it is assumed to be 78h, which maps to 120 ns or 20 clocks.
- At 200 MHz, tRFC is assumed to be 46h, which maps to 70 ns or 14 clocks for 512-Mbit or smaller devices. For 1-Gbit devices, it is assumed to be 78h, which maps to 120 ns or 24 clocks.

#### 4.1.1.8 tREF (Refresh Rate)

The tREF is not an SPD ROM parameter but a field that the DRAM controller requires and that can be obtained from several bytes in the SPD ROM. Typically, the entire DRAM must have every row refreshed at least once every 64 ms. The average time between two row refreshes is thus based on the number of rows in the DRAM and the clock frequency.

The clock frequency was previously derived. The number of rows in the implementation can be read from SPD byte 3. This provides the number of row address bits for each of the two possible chip selects on a DIMM. (Note that each chip-select range of a two chip-select DIMM uses devices that have the same number of rows). A design with 12 row address bits implies 4k rows per internal device chip select. This maps to a refresh every 15.6 µs on average. There are also 8k (13 row address bits) and 16k (14 row address bits) row devices. By knowing the number of rows and the frequency, the DRAM controller configuration register can be correctly programmed. All 4k row devices require an average refresh interval of 15.6 µs and all 8k and 16k row devices require a 7.8 µs average refresh interval.

#### 4.1.1.9 Registered or Unbuffered DIMMs

SPD byte 21, bit 1 indicates whether the DIMM registers the address and commands or not. An AMD Opteron™ processor only supports registered DIMMs and AMD Athlon™ 64 processor can support either. The AMD Athlon™ 64 processor cannot support systems in which the DIMMs are of mixed type.

#### 4.1.1.10 DIMM Chip Select Density

The size of a DIMM module chip-select range can be obtained directly from the SPD ROM.

SPD byte 5 indicates whether the DIMM is composed of one or two chip selects. Byte 31 indicates the size of the chip-select range or ranges on the DIMM (from 32 Mbyte to 2 Gbyte). See the SPD ROM specification for the encoding.

***Note:*** *If the DIMM uses two chip selects, then each chip-select range is the same size.*

#### 4.1.1.11 DIMM ECC Enable

SPD byte 11 indicates whether the DIMM supports ECC bits. A value of 02h indicates that ECC is supported for all chip-select banks on the DIMM.

#### 4.1.1.12 x4 DIMMs

The DRAM device width is largely inconsequential to the controller except when they are x4 devices. The width of the devices on the DIMM is determined by the value of SPD byte 13. If the DIMM implements two chip-select banks, then each chip-select bank uses devices of the same width. See the SPD ROM specification for the encoding.

### 4.1.2 Non-SPD ROM-Based Configuration

Many other bit fields are also required by the DRAM controller configuration registers, but these values cannot be obtained from the SPD ROM. In many cases, these values will be hardcoded into the BIOS, but in others they are functions of other bit values. This section describes how BIOS programs each non-SPD related field that is not hardcoded.

#### 4.1.2.1 Twr (Write Recovery)

This is a true DRAM device timing parameter but it is not included in the SPD ROM. Therefore, it will be hardcoded based on the DDR200, DDR266, DDR333, and DDR400 AC specification, as follows:

- DDR200, DDR266: Twr is 2 clocks.
- DDR333, DDR400: Twr is 3 clocks.

#### 4.1.2.2 Twtr (Write to Read Delay)

This parameter has the following values:

- DDR200, DDR266, DDR333: Twtr is 1 clock.
- DDR400: Twtr is 2 clocks.

#### 4.1.2.3 Trwt

This field ensures read-to-write data-bus turnaround. This field is a function of CAS latency and clock frequency. It is also a function of the asynchronous round-trip loop delay for each of the systems shown in Table 28.

**Table 28. Trwt Values**

| | | Number of Clocks | | | |
|---|---|---|---|---|---|
| **CAS Latency** | **System** | **200 MHz** | **166 MHz** | **133 MHz** | **100 MHz** |
| CL = 2 | AMD Opteron™ | N/A | 3 | 2 | 2 |
| | AMD Athlon™ 64 (registered DIMMS) | N/A | 2 | 2 | 2 |
| | AMD Athlon™ 64 (unbuffered DIMMS) | 3 | 3 | 3 | 3 |
| CL = 2.5 | AMD Opteron™ | N/A | 3 | 3 | 3 |
| | AMD Athlon™ 64 (registered DIMMS) | N/A | 3 | 3 | 3 |
| | AMD Athlon™ 64 (unbuffered DIMMS) | 4 | 4 | 4 | 4 |
| CL = 3 | AMD Opteron™ | N/A | 4 | 3 | 3 |
| | AMD Athlon™ 64 (registered DIMMS) | N/A | 3 | 3 | 3 |
| | AMD Athlon™ 64 (unbuffered DIMMS) | 4 | 4 | 4 | 4 |

#### 4.1.2.4 Twcl (Write CAS Latency)

If the DIMM is registered, the write CAS latency is two clocks. If the DIMM is unbuffered, then write CAS latency is one clock.

#### 4.1.2.5 Maximum Asynchronous Latency

AsyncLat field should be set to 9 ns for registered DIMM systems with DDR266 8-DIMM configuration, and to 8 ns for all other registered DIMM systems. AsyncLat field should be set to 7 ns for unbuffered systems with 3 DIMMs, and to 6 ns for all other unbuffered DIMM systems.

#### 4.1.2.6 Read Preamble Time

The Read Preamble time is a function of the asynchronous round trip loop latency, the clock frequency, DIMM type and process type.

**Table 29. RdPreamble Values**

| | 200 MHz | 166 MHz | 133 MHz | 100 MHz |
|---|---|---|---|---|
| Registered DIMM slots | N/A | 7.5 ns | 8 ns | 9 ns |
| Unbuffered 1 or 2 DIMM slots[1] | 5 ns | 6 ns | 7 ns | 9 ns |
| Unbuffered 3 DIMM slots[1] | 7 ns | 9 ns | 11 ns | 14 ns |

***Notes:***

1. *The values for RdPreamble must be set based on the number of DIMM slots, independent of the number of DIMMs actually populated. This allows the memory controller to compensate for worst-case round-trip delay from DIMMs in the farthest slot.*

#### 4.1.2.7 Memory Clock Enable

DRAM clocks must be enabled by BIOS. There are four clock enable bits which correspond on a one-to-one basis with the DIMMs (i.e., if DIMM0 is populated, memory clock 0 must be enabled).

DRAM clocks are grouped according to the requirements for unbuffered or registered DIMMs. For example, when MC0_EN is set to enable DIMM 0 in an unbuffered DIMMs configuration, the memory controller will drive three clock pairs to DIMM 0. In a registered DIMM configuration a single clock pair is driven to each enabled DIMM.

In DIMM configurations that have two DIMM sockets connected to the same command/address bus (as in SODIMM configurations), but have different clock pairs routed to each of the DIMMs, BIOS must enable both sets of clocks even if the DIMM 0 slot is unpopulated. For example, in an SODIMM configuration with two DIMM slots, BIOS should set both MC0_EN and MC1_EN even if only DIMM 1 is populated. This is required in order to guarantee proper operation.

### 4.1.3 DRAM Initialization

Some DRAM configuration fields must be programmed in a specific order. BIOS must set DramConfigRegLo[8] last to start DRAM initialization after a cold reset not associated with suspend to RAM mode. BIOS must set DramConfigRegLo[12] and DramConfigRegLo[13] last to exit self-refresh mode after a cold reset associated with suspend to RAM mode. BIOS should not assert LDTSTOP_L to change HyperTransport™ link width and frequency while DramConfigRegLo[8] or DramConfigRegLo[12] are set.

For Revision C, BIOS must guarantee a delay between setting DramConfigRegHi[25] and setting DramConfigRegLo[12] and DramConfigRegLo[13] when resuming from suspend to RAM mode. The required delay is 110us for registered DIMMs and 10us for unbuffered DIMMs. Both DIMM types require 10us for memory clock stabilization time. Additional 100us required for registered DIMMs is for DIMM PLL stabilization time.

## 4.2 DRAM Configuration

This section shows in a quick reference format how each DRAM controller configuration register should be programmed.

BaseAddrReg[7:0][31:0] = See "DRAM CS Base Address Registers" on page 69.

BaseMaskReg[7:0][31:0] = See "DRAM CS Mask Registers" on page 72.

BankAddrReg[31:0] = See "DRAM Bank Address Mapping Register" on page 73.

DramTimingRegLo[28] = tWR = See "Twr (Write Recovery)" on page 138.

DramTimingRegLo[26:24] = tRP[2:0] = See "tRP (Precharge Command Period)" on page 135.

DramTimingRegLo[23:20] = tRAS[3:0] = See "tRAS (Active-to-Precharge Delay)" on page 135.

DramTimingRegLo[18:16] = tRRD[2:0] = See "tRRD (Active-to-Active of a Different Bank)" on page 136.

DramTimingRegLo[14:12] = tRCD[2:0] = See "tRCD (RAS-to-CAS Delay)" on page 135.

DramTimingRegLo[11:8] = tRFC[3:0] = See "tRFC (Auto-Refresh-to-Active/Auto-Refresh Command Period)" on page 136.

DramTimingRegLo[7:4] = tRC[3:0] = See "tRC (Active-to-Active/Auto-Refresh Command Period)" on page 136.

DramTimingRegLo[2:0] = tCL[2:0] = See "tCL (CAS Latency)" on page 134.

DramTimingRegHi[22:20] = tWCL[2:0] = See "Twcl (Write CAS Latency)" on page 138.

DramTimingRegHi[12:8] = tREF[4:0] = See "tREF (Refresh Rate)" on page 136.

DramTimingRegHi[6:4] = tRWT[2:0] = See "Trwt" on page 138.

DramTimingRegHi[0] = tWTR = See "Twtr (Write to Read Delay)" on page 138.

DramConfigRegLo[27:25] = BypMax[2:0] = 4h

DramConfigRegLo[24] = DisInRcvrs = 0h[1]

DramConfigRegLo[23:20] = x4DIMMs[3:0] = See "x4 DIMMs" on page 137.

DramConfigRegLo[19] = 32ByteEn = 0h[2]

DramConfigRegLo[18] = UnBuffDimm = See "Registered or Unbuffered DIMMs" on page 137.

DramConfigRegLo[17] = DimmEccEn = See "DIMM ECC Enable,".

DramConfigRegLo[16] = 128/64 = 0h[3]

DramConfigRegLo[15:14] = RdWrQByp = 2h

DramConfigRegLo[13] = SelfRefStat = 0

DramConfigRegLo[12] = ExitSelfRef = 0h

DramConfigRegLo[8] = DramInit = 1h

DramConfigRegLo[2] = QFCEn = 0h

DramConfigRegLo[1] = DrvEn = 0h

DramConfigRegLo[0] = DllDis = 0h

DramConfigRegHi[29:26] = EnMemClk[3:0] = See "Memory Clock Enable" on page 139.

DramConfigRegHi[25] = MemClkRatioVal = 1h

DramConfigRegHi[22:20] = MemClk[2:0] = See "tCL (CAS Latency)" on page 134.

DramConfigRegHi[19] = DynIdleCtrEn = 1h

DramConfigRegHi[18:16] = IdleCycLimit[2:0] = 3h

DramConfigRegHi[11:8] = RdPreamble[3:0] = See "Read Preamble Time" on page 139.

DramConfigRegHo[3:0] = AsyncLat[3:0] = See "Maximum Asynchronous Latency" on page 139.

DramDelayLineLo[31:0] = 0000_0000h

ScrubControl[31:0] = DramScrubAddrHi[31:0] = DramScrubAddrLo[31:0] = 0000_0000h

***Notes:***

1. *Some AMD Athlon™ 64 and AMD Opteron™ processor-based systems might not use the DRAM controller. In these systems, it is preferable to disable DRAM input receivers and output drivers and to leave input receivers unconnected. This bit should be set in such systems.*
2. *This bit should be set for 64-bit interfaces (when DramConfigRegLo[16] is cleared), to enable 32-byte accesses, and cleared for 128-bit interfaces. This bit must be cleared when DramConfigRegLo[16] is set.*
3. *This bit should be cleared for 64-bit interfaces and set for 128-bit interfaces. The value of this bit is a function of the system design and cannot be determined via the SPD ROM. This value is thus hardcoded into the BIOS. AMD Athlon™ 64 systems must have this bit cleared.*

# 5 Machine Check Architecture

The AMD Athlon™ 64 processor and AMD Opteron™ processor machine check mechanism allows the processor to detect and report a variety of hardware (or machine) errors found when reading and writing data, probing, cache-line fills and writebacks. These include parity errors associated with caches and TLBs, ECC errors associated with caches and DRAM, as well as system bus errors associated with reading and writing to the external bus.

Software can enable the processor to report machine check errors through the machine check exception (See "#MC—Machine Check Exception" in *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming). Most machine check exceptions do not allow reliable restarting of the interrupted programs. However, error conditions are logged in a set of model-specific registers (MSRs) that can be used by system software to determine the possible source of a hardware problem.

## 5.1 Determining Machine Check Support

The availability of machine check registers and support of the machine check exception is implementation dependent. System software executes the CPUID instruction to determine whether a processor implements these features. After CPUID is executed, the values of the machine check architecture (MCA) bit and the machine check exception (MCE) bit loaded in the EDX register indicate whether the processor implements the machine check registers and the machine check exception, respectively. See "Processor Feature Identification" in *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming, and "CPUID" in *AMD64 Architecture Programmer's Manual*, Volume 3: General Purpose and System Instructions, for further information on the level of machine check support.

Once system software determines that the machine check registers are available, it must determine the extent of processor support for the machine check mechanism. This is accomplished by reading the machine check capabilities register (MCG_CAP). See "Machine Check Global Capabilities Register" in *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming, for more information on the interpretation of the MCG_CAP contents.

## 5.2 Machine Check Errors

Machine check errors are either recoverable or irrecoverable. Recoverable errors are those that the processor can correct and, thus, do not raise the machine check exception (#MC). However, the error is still logged in the machine check MSRs and it is the responsibility of the system software to periodically poll the machine check MSRs to determine if recoverable errors have occurred. If a recoverable error has been logged in the machine check MSRs, a second recoverable error can overwrite it.

Irrecoverable, or fatal, machine check errors cannot be corrected by the processor. If machine check is enabled, they raise the machine check exception (#MC). If an irrecoverable error has been logged in the machine check MSRs, a second recoverable or irrecoverable MCA error will not overwrite it but will set a bit that indicates overflow.

In the case of both recoverable and irrecoverable MCA errors, the contents of the machine check MSRs are maintained though a warm reset. This is helpful since in some cases it may not be possible to invoke the machine check handler due to the error and this allows the BIOS or other system boot software to recover and report the information associated with the error.

### 5.2.1 Sources of Machine Check Errors

The processor can detect errors from the following hardware blocks within the processor. Each block forms an error reporting bank for the purpose of reporting machine check errors.

- Data cache unit (DC)—Includes the cache structures that hold data and tags, the data TLBs, and the data cache probing logic.
- Instruction cache unit (IC)—Includes the instruction cache structures that hold instructions and tags, the instruction TLBs, and the instruction cache probing logic.
- Bus unit (BU)—Includes the system bus interface to the Northbridge and the level 2 cache.
- Load/store unit (LS)—Includes logic used to manage loads and stores.
- Northbridge unit (NB)—Includes the Northbridge and DRAM controller.

A scrubber is associated with the data cache in DC, the L2 cache tag array in bus unit, and the DRAM in the Northbridge. A scrubber is a hardware widget that periodically wakes up during idle cache cycles and inspects the next line of the array with which it is associated to look for errors. If it finds a single bit ECC error, the scrubber corrects the error and prevents a regular access from encountering the same error. This is important in the data cache, since all ECC errors encountered during regular operation in the data cache are fatal. In the bus unit and the Northbridge, this scrubber function also saves the additional time it would take to fix single-bit ECC errors when they are encountered during normal operation.

Even though 64-bits are read from the data cache during normal operation, a byte load will use one byte, a word load will use two bytes, and a double word load will use 4 bytes. If a data cache ECC error is detected in a byte that is not being used by a load, it is corrected like an error detected by the data cache scrubber. This feature is referred to as "piggyback scrubbing".

Table 30 on page 145 shows the various sources of machine check errors that can be encountered in the processor. ECC check on the linefill data for IC data is done by the DC unit. However, the ECC error is reported by IC which is the unit responsible for receiving the data. The data cache is protected by ECC; however single bit ECC errors encountered by normal load and store accesses are not corrected. When the data cache scrubber encounters a single-bit ECC error, it corrects it.

## Table 30. Sources of Machine Check Errors

| Unit | Error Source | Type of Check | Recoverable |
|---|---|---|---|
| DC | System line fill into the data cache | ECC | Yes—single bit error<br>No—multiple bit error |
| | L2 cache line fill into the data cache | | |
| | Cache data—data array | ECC | Yes—if single bit ECC error is detected by the scrubber<br>No—any other case different than single bit ECC error detected by the scrubber |
| | Cache data—snoop tag array | Parity | No |
| | Cache data—tag array | | No |
| | L1 Data TLB—physical and virtual arrays | | No |
| | L2 Data TLB—physical and virtual arrays | | No |
| IC | System line fill into the instruction cache | ECC | Yes—single bit error<br>No—multiple bit error |
| | L2 cache line fill into the instruction cache | | |
| | Instruction cache—data array | Parity | Yes[1] |
| | Instruction cache—tag array | | Yes[1] |
| | Instruction cache—snoop tag array | | No |
| | L1 Instruction TLB—physical and virtual arrays | | Yes[1] |
| | L2 Instruction TLB—physical and virtual arrays | | Yes[1] |
| | System address out of range | | No, but detection is precise |
| BU | L2 cache data array | ECC | Yes—single bit error<br>No—multiple bit error |
| | L2 cache—tag array (during scrubs) | | |
| | L2 cache—tag array | Parity | No.<br>Single and multiple bit errors in an L2 cache tag can also be detected, but not corrected. |
| | System Address Out-of-Range | Read Data | No |
| LS | System Address Out-of-Range | Read Data | No. Loads are detected precisely and stores are detected imprecisely. |
| NB | See "Machine Check Architecture Registers" on page 146 for information on Northbridge machine check errors. | | |

***Notes:***

1. *Instruction cache lines are invalidated and refetched.*

# 5.3 Machine Check Architecture Registers

The processor architecture defines a set of model-specific registers (MSRs) to support the MCA mechanism. They are used to configure the MCA functions and provide a way for the hardware to report errors in a manner compatible with the machine check architecture. These registers include the global MCA registers used to set up machine checks and additional banks of MSRs for recording errors that are detected by the hardware blocks listed in "Sources of Machine Check Errors" on page 144. They can be read and written using the RDMSR and WRMSR instructions. The following is a complete list of MCA MSRs.

- Global status and control registers
  - Machine check capabilities MSR (MCG_CAP)
  - Processor status MSR (MCG_STATUS)
  - Exception reporting control MSR (MCG_CTL)
- Each error reporting bank (associated with a specific hardware block listed in "Sources of Machine Check Errors" on page 144) contains the following registers:
  - Error reporting control register (MC*i*_CTL)
  - Error reporting status register (MC*i*_STATUS)
  - Error reporting address register (MC*i*_ADDR)
  - Machine check miscellaneous error information register (MC*i*_MISC)

The *i* in each register name corresponds to the number of a supported register bank. Each error-reporting register bank is associated with a specific processor unit (or group of processor units). The number of error-reporting register banks is implementation-specific.

Software reads the MCG_CAP register to determine the number of supported register banks. The AMD Athlon™ 64 and AMD Opteron™ processors support five banks. The first error-reporting register (MC0_CTL) always starts with MSR address 400h, followed by MC0_STATUS (401h), MC0_ADDR (402h), and MC0_MISC (403h). Error-reporting-register MSR addresses are assigned sequentially through the remaining supported register banks. Using this information, software can access all error-reporting registers in an implementation-independent manner.

Note that the AMD Athlon™ 64 and AMD Opteron™ processors do not support the miscellaneous error information register (MC*i*_MISC).

The global machine check registers as well as the generic form of each register in the error reporting banks are now described. The specifics of each error reporting bank will be described in "Error Reporting Banks" on page 152.

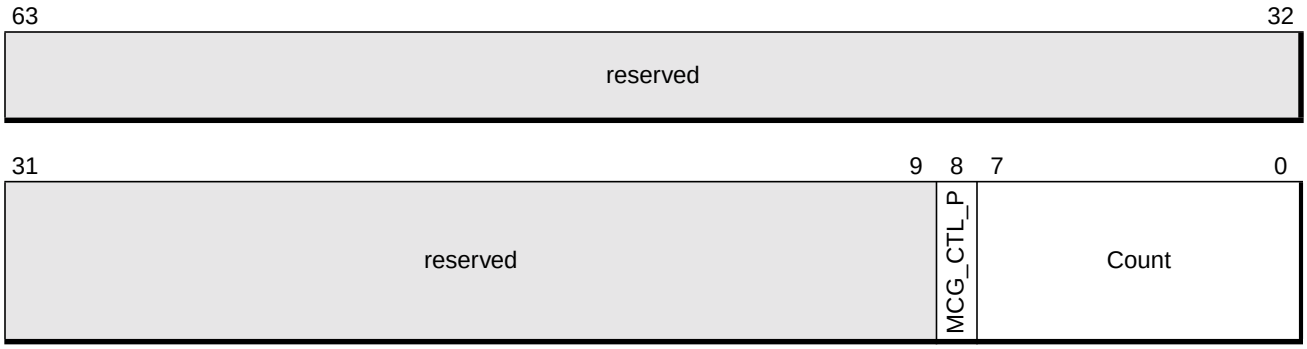
## 5.3.1 Global Machine Check Model-Specific Registers (MSRs)

The global MSRs supported by the machine check mechanism include the MCG_CAP, the MCG_STATUS and the MCG_CTL registers.

### 5.3.1.1 MCG_CAP—Global Machine Check Capabilities Register

This read-only register indicates the capabilities of the processor machine check architecture. Attempting to write to this register will result in a #GP exception. See A*MD64 Architecture Programmer's Manual, Volume 2: System Programming*, for more details.

**MCG_CAP Register** **MSR 0179h**




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–9 | reserved | | | 0 |
| 8 | MCG_CTL_P | MCG_CTL Register Present | R | 1 |
| 7–0 | Count | Count | R | 05h |

#### Field Descriptions

**Count (Count)**—Bits 7–0. Number of error-reporting banks supported by the processor implementation.

**MCG_CTL Register Present (MCG_CTL_P)**—Bit 8. Indicates if the MCG_CTL register is present. When the bit is set to 1, the register is supported. When the bit is cleared to 0, the register is unsupported.

### 5.3.1.2 MCG_STATUS—Global Machine Check Processor Status Register

This register contains basic information about the processor state after a machine check error is detected. See *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming, for more details.

**MCG_STATUS Register** **MSR 017Ah**






| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–3 | reserved | | | 0 |
| 2 | MCIP | Machine Check in Progress | R/W | 0 |
| 1 | EIPV | Error IP Valid Flag | R/W | 0 |
| 0 | RIPV | Restart IP Valid Flag | R/W | 0 |

### Field Descriptions

**Restart IP Valid Flag (RIPV)**—Bit 0. When set, indicates if program execution can be restarted at EIP pushed on stack.

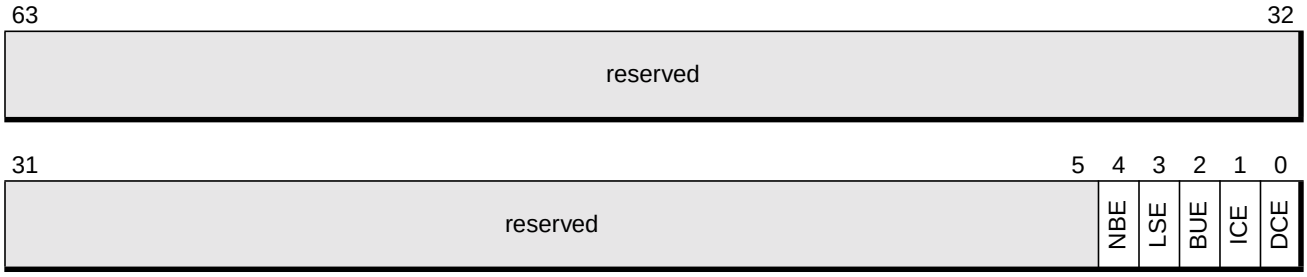
**Error IP Valid Flag (EIPV)**—Bit 1. When set, indicates if the EIP pushed on stack is that of the instruction which caused the detection of the machine check error.

**Machine Check in Progress (MCIP)**—Bit 2. When set, indicates that a machine check is in progress.

#### 5.3.1.3 MCG_CTL—Global Machine Check Exception Reporting Control Register

This register contains a global bit for each error-checking unit in the processor. Each bit enables the reporting, through the MCA interface, of errors detected by that particular unit. See *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming, for more details.

**MCG_CTL Register** **MSR 017Bh**




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–5 | reserved | | | 0 |
| 4 | NBE | NB Register Bank Enable | R/W | 0 |
| 3 | LSE | LS Register Bank Enable | R/W | 0 |
| 2 | BUE | BU Register Bank Enable | R/W | 0 |
| 1 | ICE | IC Register Bank Enable | R/W | 0 |
| 0 | DCE | DC Register Bank Enable | R/W | 0 |

### Field Descriptions

**DC Register Bank Enable (DCE)**—Bit 0. When set, indicates that the Data Cache register bank is enabled.

**IC Register Bank Enable (ICE)**—Bit 1. When set, indicates that the Instruction Cache register bank is enabled.

**BU Register Bank Enable (BUE)**—Bit 2. When set, indicates that the Bus Unit register bank is enabled.

**LS Register Bank Enable (LSE)**—Bit 3. When set, indicates that the Load/Store register bank is enabled.

**NB Register Bank Enable (NBE)**—Bit 4. When set, indicates that the Northbridge register bank is enabled.

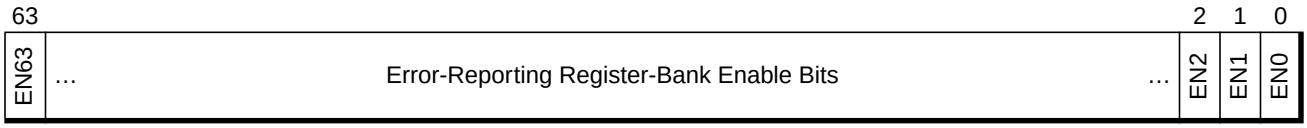
## 5.3.2 Error Reporting Bank Machine Check MSRs

The registers in each error reporting bank include MC*i*_CTL, MC*i*_STATUS and MC*i*_ADDR. The MC*i*_MISC register is not supported in AMD Athlon™ 64 and AMD Opteron™ processors.

### 5.3.2.1 MC*i*_CTL—Machine Check Control Registers

The machine check control registers (MC*i*_CTL) contain an enable bit for each error source within an error-reporting register bank. Setting an enable bit to 1 enables error-reporting for the specific feature controlled by the bit, and clearing the bit to 0 disables error reporting for the feature. These registers should generally be set to either all zeros or all ones.

**MC*i*_CTL Registers** **MSRs 0400h, 0404h, 0408h, 040Ch, 0410h**




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–0 | EN63–EN0 | Enables | R/W | 0 |

### Field Descriptions

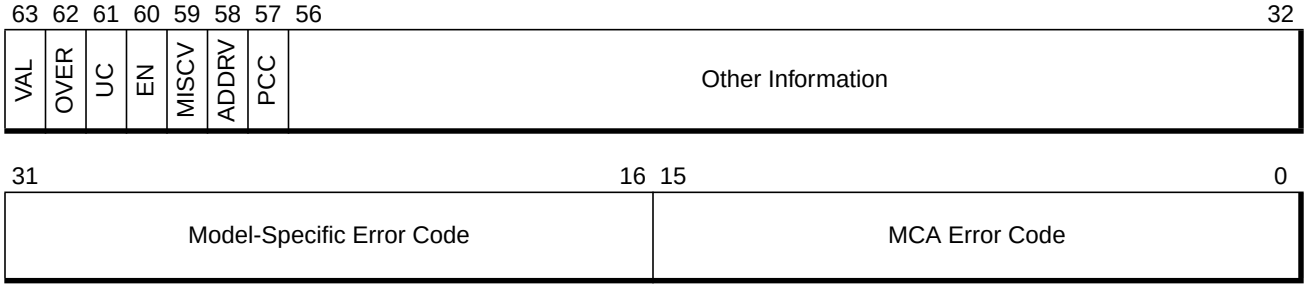
**Enables (EN63–EN0)**—Bits 63–0. If set, error reporting for the specific feature controlled by the bit is enabled. Not all these bits may be implemented in each bank.

### 5.3.2.2 MC*i*_STATUS—Machine Check Status Registers

The machine check status register (MC*i*_STATUS) describes the error detected by its unit. It is written by the processor and should be cleared to 0 by software; writing any other value to the register

causes a general protection (GP#) exception. When a machine check error occurs, the processor loads an error code into bits [15:0] of the appropriate MCi_STATUS register. Errors are classified as either system bus, cache, or TLB errors. All the status registers in each bank use the same general logging format shown below.

### MC*i*_STATUS Registers — MSRs 0401h, 0405h, 0409h, 040Dh, 0411h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63 | VAL | Valid | R/W | 0 |
| 62 | OVER | Status Register Overflow | R/W | 0 |
| 61 | UC | Uncorrected Error indication | R/W | 0 |
| 60 | EN | Error Condition Enabled | R/W | 0 |
| 59 | MISCV | Miscellaneous-Error Register Valid | R/W | 0 |
| 58 | ADDRV | Error-Address Register Valid | R/W | 0 |
| 57 | PCC | Processor-Context Corrupt | R/W | 0 |
| 56–32 | | Other Information | R/W | 0 |
| 31–16 | | Model-Specific Error Code | R/W | 0 |
| 15–0 | | MCA Error Code | R/W | 0 |

### Field Descriptions

**MCA Error Code**—Bits 15–0. This field holds an error code when an error is detected.

**Model-Specific Error Code**—Bits 31–16. This field encodes model-specific information about the error.

**Other Information**—Bits 56–32. This field holds model-specific error information.

**Processor-Context Corrupt (PCC)**—Bit 57. If set to 1, this bit indicates that the state of the processor may be corrupted by the error condition. Reliable restarting might not be possible.

0 = Processor not corrupted
1 = Processor may be corrupted

**Error-Address Register Valid (ADDRV)**—Bit 58. If set to 1, this bit indicates that the address saved in the address register is the address where the error occurred.

0 = Address register not valid
1 = Address register valid

**Miscellaneous-Error Register Valid (MISCV)**—Bit 59. If set to 1, this bit indicates whether the Miscellaneous Error register contains valid information for this error. This bit should always be 0 since the Miscellaneous Error register is not implemented.

**Error Condition Enabled (EN)**—Bit 60. If set to 1, this bit indicates that MCA error reporting is enabled for this error in the MCA Control register.

0 = Error checking not enabled
1 = Error checking enabled

**Uncorrected Error indication (UC)**—Bit 61. If set to 1, this bit indicates that the error was not corrected by hardware.

0 = Error corrected
1 = Error not corrected

**Status Register Overflow (OVER)**—Bit 62. Set to 1 if the unit detects an error but the valid bit of this register already set. Enabled errors are written over disabled errors, uncorrectable errors are written over correctable errors. Uncorrectable errors are not overwritten.

0 = No error overflow
1 = Error overflow

**Valid (VAL)**—Bit 63. If set to 1, this bit indicates that a valid error has been detected by the unit. This bit should be cleared to 0 by software after the register is read.

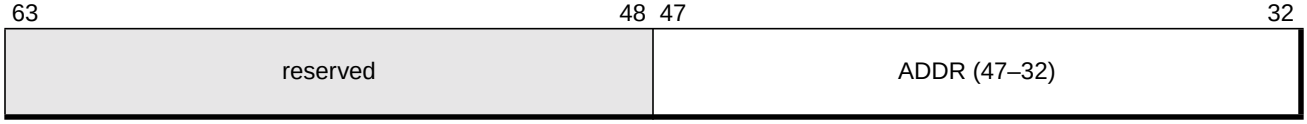
0 = No valid error detected
1 = Valid error detected

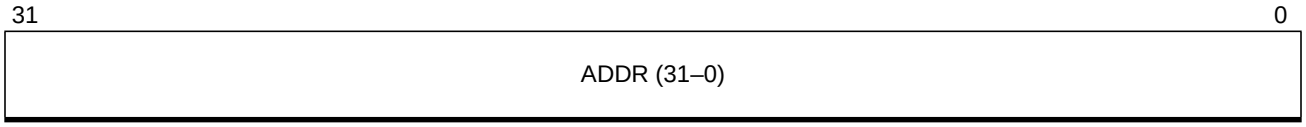
### 5.3.2.3 MC*i*_ADDR—Machine Check Address Registers

Each error-reporting register bank includes a machine check address register (MC*i*_ADDR) that the processor uses to report the instruction memory address or data memory address responsible for the machine check error. The contents of this register are valid only if the ADDRV bit in the corresponding MC*i*_STATUS register is set to 1.

The address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MC*i*_ADDR varies in width. The address field can hold either a virtual (linear) or physical address, depending on the type or error. Some error types cause the processor to load valid values into a subset of the address bits. The bit ranges depend on the values in the MC*i*_STATUS register, i.e., on the type of error detected by the unit, which can be determined by examining the MCA error code contained in the MC*i*_STATUS register.

**MC*i*_ADDR Registers** **MSRs 0402h, 0406h, 040Ah, 040Eh, 0412h**






| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–48 | reserved | | | 0 |
| 47–0 | ADDR | Address | R/W | 0 |

### Field Descriptions

**Address (ADDR)**—Bits 47–0. Not all these bits may be valid. The valid bits depend on the type of error registered in the corresponding MC*i*_STATUS register.

# 5.4 Error Reporting Banks

As mentioned in "Machine Check Architecture Registers" on page 146, the AMD Athlon™ 64 and AMD Opteron™ processors have five error-reporting banks—DC, IC, BU, LS, and NB. Each error reporting bank includes the following registers:

- Machine check control register (MC*i*_CTL).
- Machine check status register (MC*i*_STATUS).
- Machine check address register (MC*i*_ADDR).

The general format of these registers was described in "Error Reporting Bank Machine Check MSRs" on page 149. This section describes the specifics of each register as it relates to each error reporting bank.

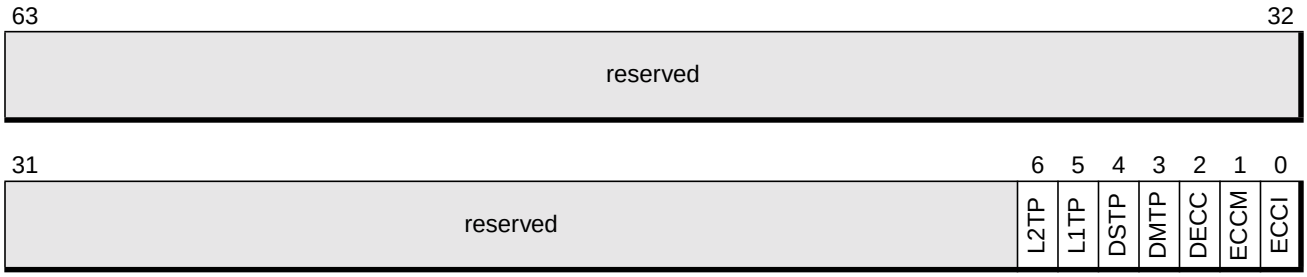
## 5.4.1 Data Cache (DC)

The data cache unit includes the level 1 data cache that holds data and tags, as well as two levels of TLBs and data cache probing logic.

### 5.4.1.1 MC0_CTL—DC Machine Check Control Register

This register enables the reporting, via the MCA interface, of a variety of errors detected by the Data Cache (DC) processor unit. For an error to be reported, both the global DC enable, MCG_CTL[DCE], and the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC0_STATUS, but not reported via the machine check exception. If the global DC enable is off, no error will be logged or reported.

## MC0_CTL Register — MSR 0400h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–7 | reserved | | | 0 |
| 6 | L2TP | L2 TLB Parity Errors | R/W | 0 |
| 5 | L1TP | L1 TLB Parity Errors | R/W | 0 |
| 4 | DSTP | Snoop Tag Array Parity Errors | R/W | 0 |
| 3 | DMTP | Main Tag Array Parity Errors | R/W | 0 |
| 2 | DECC | Data Array ECC Errors | R/W | 0 |
| 1 | ECCM | Multi-bit ECC Data Errors | R/W | 0 |
| 0 | ECCI | Single-bit ECC Data Errors | R/W | 0 |

### Field Descriptions

**Single-bit ECC Data Errors (ECCI)**—Bit 0. Report single-bit ECC data errors during data cache line fills or TLB reloads from the internal L2 or the system.

**Multi-bit ECC Data Errors (ECCM)**—Bit 1. Report multi-bit ECC data errors during data cache line fills or TLB reloads from the internal L2 or the system.

**Data Array ECC Errors (DECC)**—Bit 2. Report data cache data array ECC errors.

**Main Tag Array Parity Errors (DMTP)**—Bit 3. Report data cache main tag array parity errors.

**Snoop Tag Array Parity Errors (DSTP)**—Bit 4. Report data cache snoop tag array parity errors.

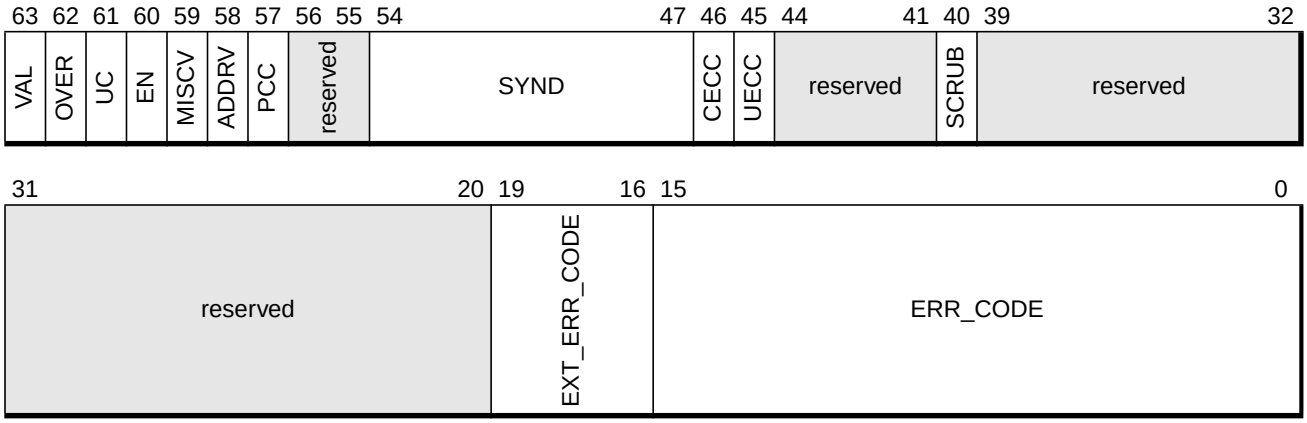
**L1 TLB Parity Errors (L1TP)**—Bit 5. Report data cache L1 TLB parity errors.

**L2 TLB Parity Errors (L2TP)**—Bit 6. Report data cache L2 TLB parity errors.

### 5.4.1.2 MC0_STATUS—DC Machine Check Status Register

The MC0_STATUS MSR describes the error that was detected by DC. The machine check mechanism writes the status register bits when an error is detected and sets the register valid bit (bit 63) to 1 to indicate that the status information is valid. This register will be written even if error reporting for the detected error is not enabled. However, error reporting must be enabled for the error to result in a machine check exception. Software is responsible for clearing this register.

## MC0_STATUS Register — MSR 0401h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63 | VAL | Valid. | R/W | 0 |
| 62 | OVER | Second error detected | R/W | 0 |
| 61 | UC | Error not corrected | R/W | 0 |
| 60 | EN | Error reporting enabled | R/W | 0 |
| 59 | MISCV | Additional info in MC*i*_MISC | R/W | 0 |
| 58 | ADDRV | Error address in MC*i*_ADDR | R/W | 0 |
| 57 | PCC | Processor state corrupted by error | R/W | 0 |
| 56–55 | reserved | | | 0 |
| 54–47 | SYND | ECC Syndrome (7–0) | R/W | 0 |
| 46 | CECC | Correctable ECC error | R/W | 0 |
| 45 | UECC | Uncorrectable ECC error | R/W | 0 |
| 44–41 | reserved | | R/W | 0 |
| 40 | SCRUB | Error detected on a scrub | R/W | 0 |
| 39–20 | reserved | | | 0 |
| 19–16 | EXT_ERR_CODE | Extended error code | R/W | 0 |
| 15–0 | ERR_CODE | Error subsection | R/W | 0 |

### Field Descriptions

**Error Subsection (ERR_CODE)**—Bits 15–0. Indicates in which subsection the error was detected and what transaction initiated it. See Table 11 on page 99 for the format of the error code and Tables 12 through 17 on pages 99–100 for descriptions of the subfields.

**Extended Error Code (EXT_ERR_CODE)**—Bits 19–16. Contains an extended error code.

*DC/IC:*

0000b = TLB parity error in physical array
0001b = TLB parity error in virtual array (multi-match error)

*BU:*

0000b = Bus or cache data array error
0010b = Cache tag array error

*LS:* Reserved

**Error Detected on a Scrub (SCRUB)**—Bit 40.

*DC:* If set, indicates that the error was detected on a scrub.

*IC/BU/LS:* Reserved.

**Uncorrectable ECC Error (UECC)**—Bit 45. If set, indicates that the error is an uncorrectable ECC error.

**Correctable ECC Error (CECC)**—Bit 46. If set, indicates that the error is a correctable ECC error.

**ECC Syndrome Bits 7–0 (SYND)**—Bits 54–47.

*DC:* The lower 8 syndrome bits when an ECC error is detected.

*IC/BU/LS:* Reserved.

**Processor State Corrupted By Error (PCC)**—Bit 57. If set, indicates that the processor state may have been corrupted by the error condition.

**Error address in MC*i*_ADDR (ADDRV)**—Bit 58. If set, indicates that the address saved in the corresponding MC*i*_ADDR register is the address where the error occurred.

**Additional Info in MC*i*_MISC (MISCV)**—Bit 59. If set, indicates that the MC*i*_MISC register contains additional info. This bit is always set to 0 on an AMD Athlon™ 64 or AMD Opteron™ processor.

**Error Reporting Enabled (EN)**—Bit 60. If set, indicates that error reporting was enabled for this error in the corresponding MC*i*_CTL register.

**Error Not Corrected (UC)**—61. If set, it indicates that the error was not corrected.

**Second Error Detected (OVER)**—Bit 62. Set if a second error is detected while the VAL bit of this register is already set.

**Valid (VAL)**—Bit 63. Set if the information in this register is valid.

#### 5.4.1.3 MC0_ADDR—DC Machine Check Address Register (MSR 0402h)

The contents of this register are valid only if the ADDRV bit in the MC0_STATUS register is set to 1. As shown in "MCi_ADDR—Machine Check Address Registers" on page 151, the address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MC0_ADDR varies in width. The bit ranges depend on the values in the MC0_STATUS register (i.e., the type of error detected) as shown in Table 31.

**Table 31. Valid MC0_ADDR Bits**

| Error Source | Access Type | Valid Address Bits |
|---|---|---|
| Snoop Tag Array | snoop | physical[11–6] |
| | victim | |
| Data Tag Array | load | physical[39–3] |
| | store | |
| Data Array | victim | physical[11–6] |
| | snoop | |
| | load | physical[39–3] |
| | store | |
| | scrub | physical[11–3] |
| L1 Data TLB | load, store | linear[47–12] |
| L2 Data TLB | | |
| L2 Cache Data | line-fill | physical[39–6] |
| System Data | | |

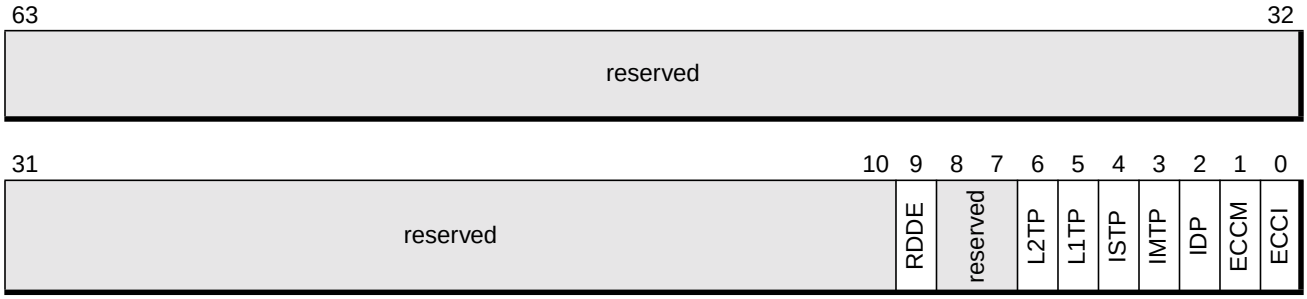
## 5.4.2 Instruction Cache (IC)

The IC unit includes the level 1 instruction cache that holds instruction data and tags, as well as two levels of TLBs and instruction cache probing logic.

### 5.4.2.1 MC1_CTL—IC Machine Check Control Register

This register enables the reporting, via the MCA interface, of a variety of errors detected by the Instruction Cache (IC) processor unit. For an error to be reported, both the global IC enable, MCG_CTL[ICE], and the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC1_STATUS, but not reported via the machine check exception. If the global IC enable is off, no error will be logged or reported.

**MC1_CTL Register** **MSR 0404h**



| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–10 | reserved | | | 0 |
| 9 | RDDE | Read Data Errors | R/W | 0 |
| 8–7 | reserved | | | 0 |
| 6 | L2TP | L2 TLB Parity Errors | R/W | 0 |
| 5 | L1TP | L1 TLB Parity Errors | R/W | 0 |
| 4 | ISTP | Snoop tag array parity errors | R/W | 0 |
| 3 | IMTP | Main tag array parity errors | R/W | 0 |
| 2 | IDP | Data array parity errors | R/W | 0 |
| 1 | ECCM | Multi-bit ECC data errors | R/W | 0 |
| 0 | ECCI | Single-bit ECC data errors | R/W | 0 |

### Field Descriptions

**Single-bit ECC Data Errors (ECCI)**—Bit 0. Report single-bit ECC data errors during instruction cache line fills or TLB reloads from the internal L2 or the system.

**Multi-bit ECC Data Errors (ECCM)**—Bit 1. Report multi-bit ECC data errors during instruction cache line fills or TLB reloads from the internal L2 or the system.

**Data Array Parity Errors (IDP)**—Bit 2. Report instruction cache data array parity errors.

**Main Tag Array Parity Errors (IMTP)**—Bit 3. Report instruction cache main tag array parity errors.

**Snoop Tag Array Parity Errors (ISTP)**—Bit 4. Report instruction cache snoop tag array parity errors.

**L1 TLB Parity Errors (L1TP)**—Bit 5. Report instruction cache L1 TLB parity errors.

**L2 TLB Parity Errors (L2TP)**—Bit 6. Report instruction cache L2 TLB parity errors.

**Read Data Errors (RDDE)**—Bit 9. Report system read data errors for an instruction cache fetch if MC2_CTL[S_RDE_ALL] = 1.

#### 5.4.2.2 MC1_STATUS—IC Machine Check Status Register (MSR 0405h)

The MC1_STATUS MSR describes the error that was detected by IC and is very similar to MC0_STATUS. See "MC0_STATUS—DC Machine Check Status Register" on page 153 for a description of the fields in this register.

#### 5.4.2.3 MC1_ADDR—IC Machine Check Address Register (MSR 0406h)

The contents of this register are valid only if the ADDRV bit in the MC1_STATUS register is set to 1. The address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MC1_ADDR varies in width. The bits ranges depend on the values in the MC1_STATUS register (i.e., the type of error detected) and this is shown in Table 32.

**Table 32. Valid MC1_ADDR Bits**

| Error Source | Access Type | Valid Address Bits |
|---|---|---|
| Snoop Tag Array | Snoop | Physical[39–6] |
| | Victim | None |
| Instruction Tag Array | Code Read | Linear[47–6] |
| | Victim | None |
| Instruction Data Array | Code Read | Linear[47–4] |
| L1 TLB | Code Read | Linear[47–12] for 4-Kbyte page<br>Linear[47–20] for 2-Mbyte page |
| L2 TLB | Code Read | Linear[47–12] for 4-Kbyte page |
| L2 Cache Data | Line-fill | Physical[39–6] |
| System Data | | |
| System Address Out of Range | | None |

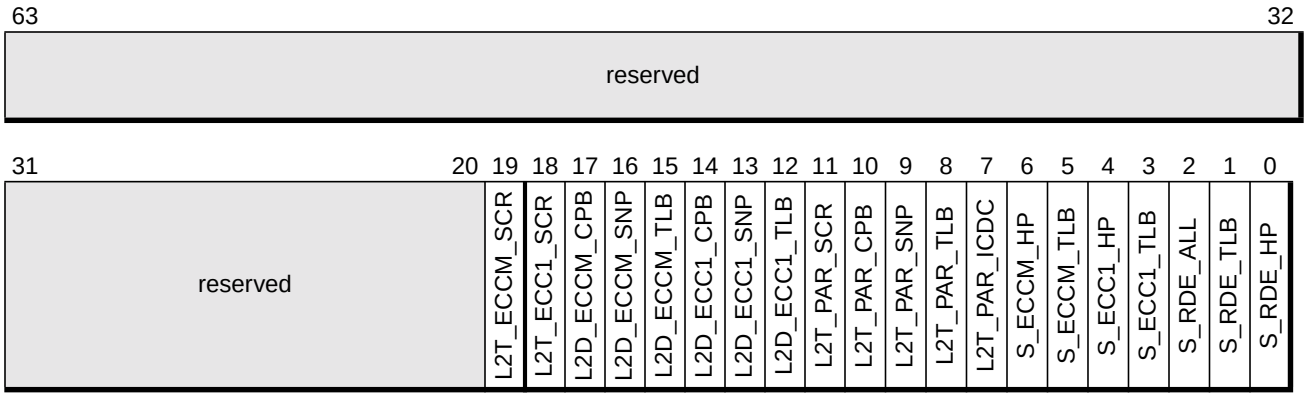
## 5.4.3 Bus Unit (BU)

The bus unit consists of the system bus interface logic and the L2 cache.

### 5.4.3.1 MC2_CTL—BU Machine Check Control Register

This register enables the reporting, via the MCA interface, of a variety of errors detected in the Bus processor unit (BU). For an error to be reported, both the global BU enable, MCG_CTL[BUE], and the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC2_STATUS, but not reported via the machine check exception. If the global BU enable is off, no error will be logged or reported.

**MC2_CTL Register** **MSR 0408h**



| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–20 | reserved | | | 0 |
| 19 | L2T_ECCM_SCR | L2 tag array multi-bit ECC Scrub | R/W | 0 |
| 18 | L2T_ECC1_SCR | L2 tag array 1-bit ECC Scrub | R/W | 0 |
| 17 | L2D_ECCM_CPB | L2 data array multi-bit ECC copyback | R/W | 0 |
| 16 | L2D_ECCM_SNP | L2 data array multi-bit ECC snoop | R/W | 0 |
| 15 | L2D_ECCM_TLB | L2 data array multi-bit ECC TLB reload | R/W | 0 |
| 14 | L2D_ECC1_CPB | L2 data array 1-bit ECC copyback | R/W | 0 |
| 13 | L2D_ECC1_SNP | L2 data array 1-bit ECC snoop | R/W | 0 |
| 12 | L2D_ECC1_TLB | L2 data array 1-bit ECC TLB reload | R/W | 0 |
| 11 | L2T_PAR_SCR | L2 tag array parity scrub | R/W | 0 |
| 10 | L2T_PAR_CPB | L2 tag array parity copyback | R/W | 0 |
| 9 | L2T_PAR_SNP | L2 tag array parity snoop | R/W | 0 |
| 8 | L2T_PAR_TLB | L2 tag array parity TLB reload | R/W | 0 |
| 7 | L2T_PAR_ICDC | L2 tag array parity IC or DC fetch | R/W | 0 |
| 6 | S_ECCM_HP | System data multi-bit ECC hardware prefetch | R/W | 0 |
| 5 | S_ECCM_TLB | System data multi-bit ECC TLB reload | R/W | 0 |
| 4 | S_ECC1_HP | System data 1-bit ECC hardware prefetch | R/W | 0 |
| 3 | S_ECC1_TLB | System data 1-bit ECC TLB reload | R/W | 0 |
| 2 | S_RDE_ALL | All system read data | R/W | 0 |
| 1 | S_RDE_TLB | System read data TLB reload | R/W | 0 |
| 0 | S_RDE_HP | System read data hardware prefetch | R/W | 0 |

## Field Descriptions

**System Read Data Hardware Prefetch (S_RDE_HP)**—Bit 0. Report system read data errors for a hardware prefetch.

**System Read Data TLB Reload (S_RDE_TLB)**—Bit 1. Report system read data errors for a TLB reload.

**All System Read Data (S_RDE_ALL)**—Bit 2. Report system read data errors for any operation including a DC/IC fetch, TLB reload or hardware prefetch.

**System Data 1-bit ECC TLB Reload (S_ECC1_TLB)**—Bit 3. Report system data 1-bit ECC errors for a TLB reload.

**System Data 1-bit ECC Hardware Prefetch (S_ECC1_HP)**—Bit 4. Report system data 1-bit ECC errors for a hardware prefetch.

**System Data Multi-bit ECC TLB Reload (S_ECCM_TLB)**—Bit 5. Report system data multi-bit ECC errors for a TLB reload.

**System Data Multi-bit ECC Hardware Prefetch (S_ECCM_HP)**—Bit 6. Report system data multi-bit ECC errors for a hardware prefetch.

**L2 Tag Array Parity IC or DC Fetch (L2T_PAR_ICDC)**—Bit 7. Report L2 tag array parity errors for an IC or DC fetch.

**L2 Tag Array Parity TLB Reload (L2T_PAR_TLB)**—Bit 8. Report L2 tag array parity errors for a TLB reload.

**L2 Tag Array Parity Snoop (L2T_PAR_SNP)**—Bit 9. Report L2 tag array parity errors for a snoop.

**L2 Tag Array Parity Copyback (L2T_PAR_CPB)**—Bit 10. Report L2 tag array parity errors for a copyback.

**L2 Tag Array Parity Scrub (L2T_PAR_SCR)**—Bit 11. Report L2 tag array parity errors for a scrub.

**L2 Data Array 1-bit ECC TLB Reload (L2D_ECC1_TLB)**—Bit 12. Report L2 data array 1-bit ECC errors for a TLB reload.

**L2 Data Array 1-bit ECC Snoop (L2D_ECC1_SNP)**—Bit 13. Report L2 data array 1-bit ECC errors for a snoop.

**L2 Data Array 1-bit ECC Copyback (L2D_ECC1_CPB)**—Bit 14. Report L2 data array 1-bit ECC errors for a copyback.

**L2 Data Array Multi-bit ECC TLB Reload (L2D_ECCM_TLB)**—Bit 15. Report L2 data array multi-bit ECC errors for a TLB reload.

**L2 Data Array Multi-bit ECC Snoop (L2D_ECCM_SNP)**—Bit 16. Report L2 data array multi-bit ECC errors for a snoop.

**L2 Data Array Multi-bit ECC Copyback (L2D_ECCM_CPB)**—Bit 17. Report L2 data array multi-bit ECC errors for a copyback.

**L2 Tag Array 1-bit ECC Scrub (L2T_ECC1_SCR)**—Bit 18. Report L2 tag array 1-bit ECC errors for a scrub.

**L2 Tag Array Multi-bit ECC Scrub (L2T_ECCM_SCR)**—Bit 19. Report L2 tag array multi-bit ECC errors for a scrub.

#### 5.4.3.2 MC2_STATUS—BU Machine Check Status Register (MSR 0409h)

The MC2_STATUS MSR describes the error that was detected by BU and is very similar to MC0_STATUS. See "MC0_STATUS—DC Machine Check Status Register" on page 153 for a description of the fields in this register.

#### 5.4.3.3 MC2_ADDR—BU Machine Check Address Register (MSR 040Ah)

The contents of this register are valid only if the ADDRV bit in the MC2_STATUS register is set to 1. The address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MC2_ADDR varies in width. The bit ranges depend on the values in the MC2_STATUS register (i.e., the type of error detected) and this is shown in Table 33.

**Table 33. Valid MC2_ADDR Bits**

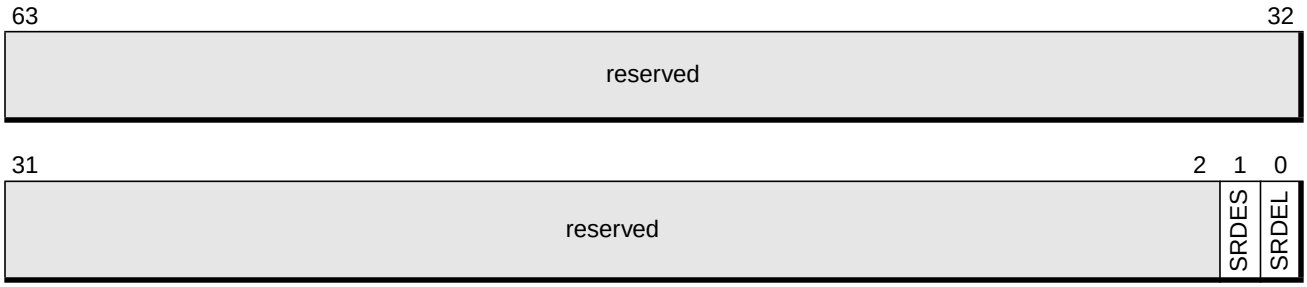
| Error Source | Access Type | Valid Address Bits |
|---|---|---|
| L2 Cache—Data Array | Any | Physical[39–6] |
| L2 Cache—Tag Array | Any | MC2_ADDR[3–0] contains encoded cache way<br>Physical[15–6] for 1-Mbyte L2<br>Physical[14–6] for 512-Kbyte L2<br>Physical[13–6] for 256-Kbyte L2<br>Physical[12–6] for 128-Kbyte L2 |
| System Address Out of Range | Any | Physical[39–6] |

## 5.4.4 Load Store Unit (LS)

### 5.4.4.1 MC3_CTL—LS Machine Check Control Register

This register enables the reporting, via the MCA interface, of two types of errors detected by the Load/Store (LS) processor unit. For an error to be reported, both the global LS enable, MCG_CTL[LSE], and the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC3_STATUS, but not reported via the machine check exception. If the global LS enable is off, no error will be logged or reported.

**MC3_CTL Register** **MSR 040Ch**




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–2 | reserved | | | 0 |
| 1 | S_RDE_S | Read Data Errors on Store | R/W | 0 |
| 0 | S_RDE_L | Read Data Errors on Load | R/W | 0 |

**Field Descriptions**

**Read Data Errors on Load (S_RDE_L)**—Bit 0. Report system read data errors on a load if MC2_CTL[S_RDE_ALL] = 1.

**Read Data Errors on Store (S_RDE_S)**—Bit 1. Report system read data errors on a store if MC2_CTL[S_RDE_ALL] = 1.

#### 5.4.4.2 MC3_STATUS—LS Machine Check Status Register (MSR 040Dh)

The MC3_STATUS MSR describes the error that was detected by LS and is very similar to MC0_STATUS. See "MC0_STATUS—DC Machine Check Status Register" on page 153 for a description of the fields in this register.

#### 5.4.4.3 MC3_ADDR—LS Machine Check Address Register (MSR 040Eh)

The contents of this register are valid only if the ADDRV bit in the MC3_STATUS register is set to 1. The only type of error recorded by the LS machine check mechanism is a "system address out of range" or read data error for which MC3_ADDR[39:0] store the physical address.

### 5.4.5 Northbridge (NB)

The Northbridge and DRAM memory controller are included in this block.

#### 5.4.5.1 MC4_CTL—NB Machine Check Control Register (MSR 0410h)

This register enables the reporting, via the MCA interface, of the errors detected by the North Bridge (NB) processor unit. "MCA NB Control Register" (Function 3, Offset 40h) maps to MC4_CTL[31:0]. See "MCA NB Control Register" on page 92 for detailed information on this register. For an error to be reported, both the global NB enable, MCG_CTL[NBE], and the corresponding local enable must be set. If the local enable is off, the error will only be logged in MC4_STATUS, but not reported via the machine check exception. If the global NB enable is off, no error will be logged or reported.

#### 5.4.5.2 MC4_STATUS—NB Machine Check Status Register (MSR 0411h)

The MC4_STATUS MSR describes the error that was detected by the Northbridge. "MCA NB Status Low Register" (Function 3, Offset 48h) maps to MC4_STATUS MSR[31:0] and "MCA NB Status High Register" (Function 3, Offset 4Ch) maps to MC4_STATUS MSR[63:32]. See "MCA NB Status Low Register" on page 98 and "MCA NB Status High Register" on page 101 for more detailed information.

#### 5.4.5.3 MC4_ADDR—NB Machine Check Address Register (MSR 0412h)

The contents of this register are valid only if the ErrAddrVal bit in the MC4_STATUS register is set to 1. "MCA NB Address Low Register" (Function 3, Offset 50h) maps to MC4_ADDR[31:0] and "MCA NB Address High Register" (Function 3, Offset 54h) maps to MC4_ADDR[63:32]. See "MCA NB Address Low Register" on page 103 and "MCA NB Address High Register" on page 104 for a description of this register.

## 5.5 Initializing the Machine Check Mechanism

Following is the general process system software should follow to initialize the machine check mechanism:

1. Execute the CPUID and verify that the processor supports the machine check exception (MCE) and MCA. MCE is supported when EDX bit 7 is set to 1, and MCA is supported when EDX bit 14 set to 1. Software should not proceed with initializing the machine check mechanism if MCE is not supported.
2. If MCA is supported, system software should take the following steps:
   a. Check to see if the MCG_CTL_P bit in the MCG_CAP register is set to 1. If it is, then the MCG_CTL register is supported by the processor. When this register is supported, software should set its enable bits to 1 for the machine check features it uses. Software can load MCG_CTL with all 1s to enable all machine check features.
   b. Read the COUNT field from the MCG_CAP register to determine the number of error-reporting register banks supported by the processor. This is set to 5 in AMD Athlon™ 64 and AMD Opteron™ processors since there are five blocks—DC, IC, BU, LS, and NB. For each error-reporting register bank, software should set the enable bits to 1 in the MC*i*_CTL register for the error types it wants the processor to report. Software can load each MC*i*_CTL register bit with a 1 to enable all error-reporting mechanisms.

      The error-reporting register banks are numbered from 0 to one less than the value found in the MCG_CAP.COUNT field. For example, when the COUNT field indicates that 5 register banks are supported, they are numbered 0 to 4.
   c. For each error-reporting register bank, software should clear all status fields in the MC*i*_STATUS register by writing all 0s to the register.

      It is possible that valid error status is reported in the MC*i*_STATUS registers at the time software clears them. The status can reflect fatal errors recorded before a processor reset or errors recorded during the system power-up and boot process. Prior to clearing the MC*i*_STATUS registers, software should examine their contents and log any errors found.
3. As a final step in the initialization process, system software should enable the machine check exception by setting CR4.MCE (bit 6) to 1.

## 5.6 Using Machine Check Features

System software can detect and handle machine check errors using two methods:

- Software can periodically examine the machine check status registers for reported errors, and log any errors found.
- Software can enable the machine check exception (#MC). When an uncorrectable error occurs, the processor immediately transfers control to the machine check exception handler. In this case, system software provides a machine check exception handler that, at a minimum, logs detected

errors. The exception handler can be designed for a specific processor implementation or can be generalized to work on multiple implementations.

### 5.6.1 Handling Machine Check Exceptions

The processor uses the interrupt control-transfer mechanism to invoke an exception handler after a machine check exception occurs. This requires system software to initialize the interrupt-descriptor table (IDT) with either an interrupt gate or a trap gate that references the interrupt handler.

At a minimum, the machine check exception handler must be capable of logging errors for later examination. Because most machine check errors are not recoverable, the ability to log errors can be sufficient for implementation of the handler. More thorough exception-handler implementations can analyze errors to determine if each error is recoverable. If a recoverable error is identified, the exception handler can attempt to correct the error and restart the interrupted program.

Machine check exception handlers that attempt to correct recoverable errors must be thorough in their analysis and the corrective actions they take. The following guidelines should be used when writing such a handler:

- All status registers in the error-reporting register banks must be examined to identify the cause or causes of the machine check exception. Read the COUNT field from MCG_CAP to determine the number of status registers supported by the processor. The status registers are numbered from 0 to one less than the value found in the MCG_CAP.COUNT field. For example, if the COUNT field indicates five status registers are supported, they are numbered MC0_STATUS to MC4_STATUS.
- Check the valid bit in each status register (MC*i*_STATUS.VAL). The MC*i*_STATUS register does not need to be examined when its valid bit is clear.
- Check the valid MC*i*_STATUS registers to see if error recovery is possible. Error recovery is not possible when:
  - The processor-context corrupt bit (MC*i*_STATUS.PCC) is set to 1.
  - The error-overflow status bit (MC*i*_STATUS. OVER) is set to 1. This bit indicates that more than one machine check error has occurred, but only one error is reported by the status register.

If error recovery is not possible, the handler should log the error information and return to the operating system.

- Check the MC*i*_STATUS.UC bit to see if the processor corrected the error. If UC=1, the processor did not correct the error, and the exception handler must correct the error prior to restarting the interrupted program. If the handler cannot correct the error, it should log the error information and return to the operating system.
- When identifying the error condition, portable exception handlers should examine only the MCA error-code field of the MC*i*_STATUS register. See the error codes in tables 12 through 17 for information on interpreting this field.

If the MCG_STATUS.RIPV bit is set to 1, the interrupted program can be restarted reliably at the instruction-pointer address pushed onto the exception-handler stack. If RIPV = 0, the interrupted program cannot be restarted reliably, although it may be possible to restart it for debugging purposes.

- When logging errors, particularly those that are not recoverable, check the MCG_STATUS.EIPV bit to see if the instruction-pointer address pushed onto the exception-handler stack is related to the machine check error. If EIPV = 0, the address is not guaranteed to be related to the error.
- Prior to exiting the machine check handler, be sure to clear MCG_STATUS.MCIP to 0. MCIP indicates that a machine check exception occurred. If this bit is set when another machine check exception occurs, the processor enters the shutdown state.
- When an exception handler is able to, at a minimum, successfully log an error condition, clear the MC*i*_STATUS registers prior to exiting the machine check handler. Software is responsible for clearing at least the MC*i*_STATUS.VAL bit.
- Additional machine check exception-handler portability can be added by having the handler use the CPUID instruction to identify the processor and its capabilities. Implementation-specific software can be added to the machine check exception handler based on the processor information reported by CPUID.

#### 5.6.1.1 Reporting Correctable Machine Check Errors

Machine check exceptions do not occur if the error is correctable by the processor. If system software wishes to log and report correctable machine check errors, a system-service routine must be provided to check the contents of the machine check status registers for correctable errors. The service routine can be invoked by system software on a periodic basis, or it can be manually invoked by the user as needed.

Assuming that the processor supports the machine check registers, a service routine that reports correctable errors should perform the following:

1. Examine each status register (MC*i*_STATUS) in the error-reporting register banks. For each MC*i*_STATUS register with a set valid bit (VAL = 1), the service routine should:
   a. Save the contents of the MC*i*_STATUS register.
   b. Save the contents of the corresponding MC*i*_ADDR register if MC*i*_STATUS.ADDRV = 1.
   c. Save the contents of the corresponding MC*i*_MISC register if MC*i*_STATUS.MISCV = 1.
   d. Check to see if MCG_STATUS.MCIP = 1, indicating that the machine check exception handler is in progress. If this is the case, then the machine check exception handler has called the service routine to log the errors. In this situation, the error-logging service routine should determine whether or not the interrupted program is restartable, and report the determination back to the exception handler. The program is *not restartable* if either of the following is true:
      – MC*i*_STATUS.PCC = 1, indicating the processor context is corrupted.
      – MCG_STATUS.RIPV = 0, indicating the interrupted program cannot be restarted reliably at the instruction-pointer address pushed onto the exception-handler stack.

2. Once the information found in the error-reporting register banks is saved, the MC*i*_STATUS register should be cleared to 0. This allows the processor to properly report any subsequent errors in the MC*i*_STATUS registers.
3. In multiprocessor configurations, the service routine can save the processor node identifier. This can help locate a failing multiprocessor-system component, which can then be isolated from the rest of the system.

# 6 System Management Mode (SMM)

System management mode (SMM) is an operating mode entered when system management interrupt SMI# is asserted. The SMI is handled by a dedicated interrupt handler pointed to by processor model specific registers. SMM is used for system control activities such as power management. These activities are transparent to conventional operating systems (such as MS-DOS and Windows® operating system). SMM is used by the BIOS and specialized low level device drivers. The code and data for SMM are stored in the SMM memory area, which may be isolated from the main memory accesses using special processor functions.

This chapter describes the SMM state save area, entry into and exit from SMM, exceptions and interrupts in SMM, and memory allocation and addressing in SMM.

## 6.1 SMM Overview

The processor enters SMM after the system logic asserts the SMI# interrupt and the processor recognizes SMI# active. The processor may be programmed to send a special bus cycle to the system, indicating that it is entering SMM mode. The processor saves its state into the SMM memory state save area and jumps to the SMM service routine. The processor returns from SMM by executing the RSM instruction in the SMM service routine. The processor restores its state from the SMM state save area and resumes execution of the instruction following the point where it entered SMM. The processor may be programmed to send a special bus cycle to the system, indicating that it is exiting SMM mode.

## 6.2 Operating Mode and Default Register Values

The software environment after entering SMM mode has the following characteristics:

- Addressing and operation in Real mode.
- 4-Gbyte segment limits.
- Default 16-bit operand, address, and stack sizes (instruction prefixes can override these defaults).
- Control transfers that do not override the default operand size truncate the EIP to 16 bits.
- Far jumps or calls cannot transfer control to a segment with a base address requiring more than 20 bits, like in Real mode segment-base addressing, unless a change is made into protected mode
- A20M# is disabled. A20M# assertion or deassertion have no effect on SMM mode.
- Interrupt vectors use the Real mode interrupt vector table.
- The IF flag in EFLAGS is cleared (INTR is not recognized).

- The TF flag in EFLAGS is cleared.
- The NMI and INIT interrupts are masked.
- Debug register DR7 is cleared (debug traps are disabled).

Figure 2 shows the default map of the SMM memory area. It consists of a 64-Kbyte area, between 0003_0000h and 0003_FFFFh. The top 32 Kbytes (0003_8000–0003_FFFFh) must be populated with RAM. The default code-segment (CS) base address for the area (called the SMM_BASE address) is 0003_0000h. The top 512 bytes (0003_FE00–0003_FFFFh) are the SMM state save area. The default entry point for the SMM service routine is 0003_8000h.

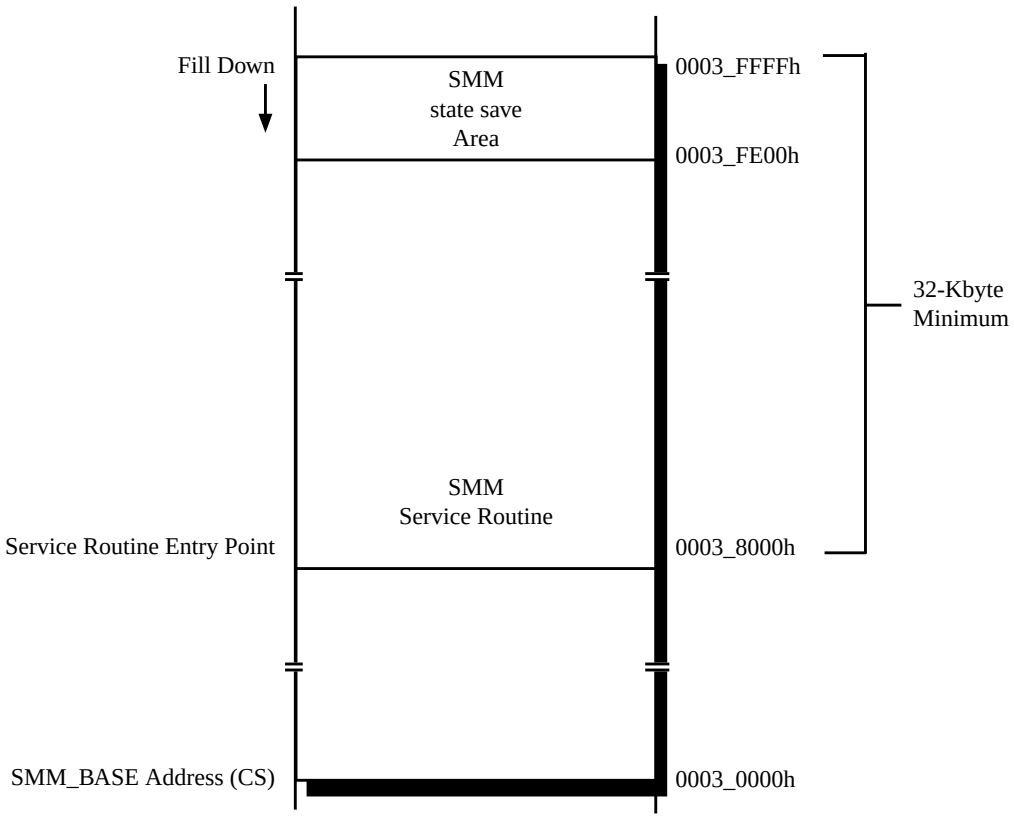



**Figure 2. Default SMM Memory Map**

# 6.3 SMM State Save Definition

**Table 34. SMM Save State (Offset FE00–FFFFh)**

| Offset (Hex) from SMM_BASE[1] | Contents | | Size | Allowable Access |
|---|---|---|---|---|
| FE00h | ES | Selector | Word | Read-Only |
| FE02h | | Attributes | Word | |
| FE04h | | Limit | Doubleword | |
| FE08h | | Base | Quadword | |
| FE10h | CS | Selector | Word | Read-Only |
| FE12h | | Attributes | Word | |
| FE14h | | Limit | Doubleword | |
| FE18h | | Base | Quadword | |
| FE20h | SS | Selector | Word | Read-Only |
| FE22h | | Attributes | Word | |
| FE24h | | Limit | Doubleword | |
| FE28h | | Base | Quadword | |
| FE30h | DS | Selector | Word | Read-Only |
| FE32h | | Attributes | Word | |
| FE34h | | Limit | Doubleword | |
| FE38h | | Base | Quadword | |
| FE40h | FS | Selector | Word | Read-Only |
| FE42h | | Attributes | Word | |
| FE44h | | Limit | Doubleword | |
| FE48h | | Base | Quadword | |
| FE50h | GS | Selector | Word | Read-Only |
| FE52h | | Attributes | Word | |
| FE54h | | Limit | Doubleword | |
| FE58h | | Base | Quadword | |
| FE60h–FE61h | GDTR | reserved | 2 Bytes | Read-Only |
| FE62h | | reserved | Word | |
| FE64h | | Limit | Word | |
| FE66h–FE67h | | reserved | 2 Bytes | |
| FE68h | | Base | Quadword | |

***Notes:***

1. *The offset for the SMM-revision identifier is compatible with previous implementations.*

## Table 34. SMM Save State (Offset FE00–FFFFh) (Continued)

| Offset (Hex) from SMM_BASE[1] | Contents | | Size | Allowable Access |
|---|---|---|---|---|
| FE70h | LDTR | Selector | Word | Read-Only |
| FE72h | | Attributes | Word | |
| FE74h | | Limit | Doubleword | |
| FE78h | | Base | Quadword | |
| FE80h–FEB1h | IDTR | reserved | 2 Bytes | Read-Only |
| FE82h | | reserved | Word | |
| FE84h | | Limit | Word | |
| FEB6h–FEB7h | | reserved | 2 Bytes | |
| FE88h | | Base | Quadword | |
| FE90h | TR | Selector | Word | Read-Only |
| FE92h | | Attributes | Word | |
| FE94h | | Limit | Doubleword | |
| FE98h | | Base | Quadword | |
| FEA0h–FEBFh | reserved | | 32 Bytes | — |
| FEC0h–FEC3h | SMM I/O Trap | | Doubleword | Read-Only |
| FEC4h–FEC7h | reserved | | 4 Bytes | — |
| FEC8h | I/O Instruction Restart | | Byte | Read/Write |
| FEC9h | Auto-Halt Restart | | Byte | |
| FECAh—FECFh | reserved | | 6 Bytes | — |
| FED0h | EFER | | Quadword | Read-Only |
| FED8h—FEFBh | reserved | | 36 Bytes | — |
| FEFCh | SMM-Revision Identifier[1] | | Doubleword | Read-Only |
| FF00h | SMM_BASE | | Doubleword | Read/Write |
| FF04h—FF47h | reserved | | 68 Bytes | — |
| FF48h | CR4 | | Quadword | Read-Only |
| FF50h | CR3 | | Quadword | |
| FF58h | CR0 | | Quadword | |
| FF60h | DR7 | | Quadword | Read-Only |
| FF68h | DR6 | | Quadword | |
| FF70h | RFLAGS | | Quadword | Read/Write |

***Notes:***

1. *The offset for the SMM-revision identifier is compatible with previous implementations.*

**Table 34. SMM Save State (Offset FE00–FFFFh) (Continued)**

| Offset (Hex) from SMM_BASE[1] | Contents | Size | Allowable Access |
|---|---|---|---|
| FF78h | RIP | Quadword | Read/Write |
| FF80h | R15 | Quadword | |
| FF88h | R14 | Quadword | |
| FF90h | R13 | Quadword | |
| FF98h | R12 | Quadword | |
| FFA0h | R11 | Quadword | |
| FFA8h | R10 | Quadword | |
| FFB0h | R9 | Quadword | |
| FFB8h | R8 | Quadword | |
| FFC0h | RDI | Quadword | Read/Write |
| FFC8h | RSI | Quadword | |
| FFD0h | RBP | Quadword | |
| FFD8h | RSP | Quadword | |
| FFE0h | RBX | Quadword | |
| FFE8h | RDX | Quadword | |
| FFF0h | RCX | Quadword | |
| FFF8h | RAX | Quadword | |

***Notes:***
1. *The offset for the SMM-revision identifier is compatible with previous implementations.*

After recognizing the SMI# assertion, the processor saves almost the entire integer processor state to memory. Exceptions are: the floating point state, the model specific registers, and CR2. Any processor state not saved in the save state must be saved and restored by the SMM handler. With the AMD Athlon™ 64 processor and AMD Opteron™ processor extension of register size to 64-bits, the SMM save state has changed considerably compared to the legacy 32-bit version. One exception is the SMM Revision Identifier, which is located in the same position as before, for proper SMM identification.

***Note:*** *The save state will always be 64-bit, regardless of the operating mode (32-bit or 64-bit).*

# 6.4 SMM Initial State

After storing the save state, the processor starts executing the handler at address SMM_BASE + 08000h. It starts with the entry state listed in Table 35.

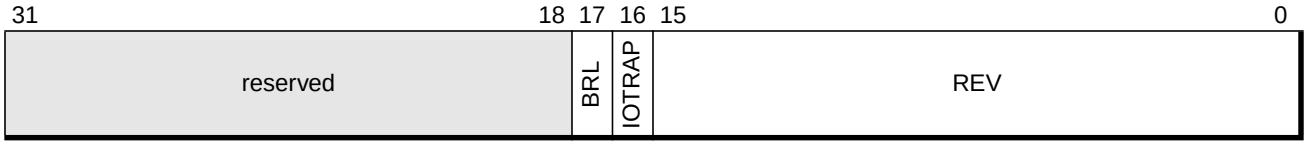
**Table 35. SMM Entry State**

| Register | Initial Contents | | |
|---|---|---|---|
| | Selector | Base | Limit |
| CS | SMM_BASE[19:4] | SMM_BASE[31:0] | 4 Gbytes |
| DS | 0000h | 0000_0000h | 4 Gbytes |
| ES | 0000h | 0000_0000h | 4 Gbytes |
| FS | 0000h | 0000_0000h | 4 Gbytes |
| GS | 0000h | 0000_0000h | 4 Gbytes |
| SS | 0000h | 0000_0000h | 4 Gbytes |
| General-Purpose Registers | Unmodified | | |
| EFLAGS | 0000_0002h | | |
| EIP | 0000_8000h | | |
| CR0 | Bits 0, 2, 3, and 31 cleared (PE, EM, TS, and PG); remainder is unmodified | | |
| CR4 | 0000_0000h | | |
| GDTR | Unmodified | | |
| LDTR | Unmodified | | |
| IDTR | Unmodified | | |
| TR | Unmodified | | |
| DR7 | 0000_0400h | | |
| DR6 | Undefined | | |

# 6.5 SMM-Revision Identifier

The SMM-Revision Identifier specifies the version of SMM and the processor extensions.

**SMM-Revision Identifier** **Offset FEFCh**




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–18 | reserved | SBZ | R | |
| 17 | BRL | SMM Base Relocation Supported | R | 1 |
| 16 | IOTRAP | SMM I/O Trap Supported | R | 1 |
| 15–0 | REV | SMM Revision Level | R | 0064h |

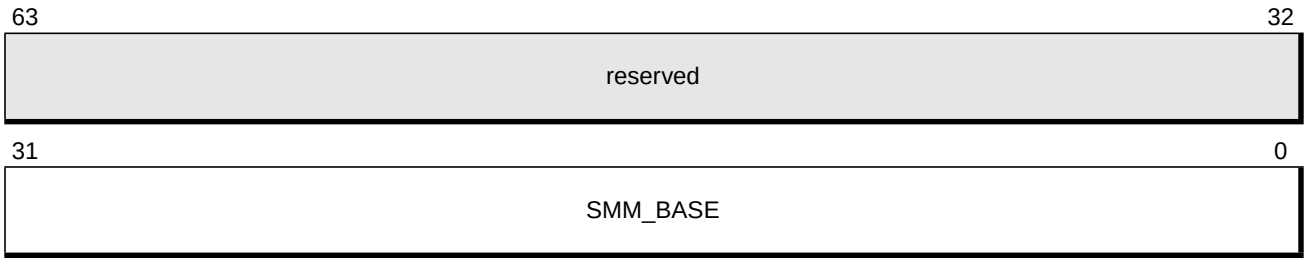
# 6.6 SMM Base Address

SMM_BASE register holds the base of the system management memory region, and its default value is 30000h. The value of this register is stored in the save state on entry into SMM and it is restored on returning from SMM. The first SMI handler instruction is fetched at SMM_BASE + 8000h. The 16 bit code segment selector is formed on entering SMM from SMM_BASE[19–4]. SMM_BASE[31–20] and SMM_BASE[3–0] have to be 0; if not, the CS base will not be equal to the (CS selector << 4), and attempts to load the CS_BASE into other descriptors will not work properly, since the selector cannot properly represent the base. If there is a far branch in the SMM handler, it will only be able to address the lower 1M of memory and will not be able to restore the SMM base to a value above 1M, unless it first switches to protected mode.

The SMM base address can be changed in two ways:

- The SMM base address, at offset FF00h in the SMM state save area, can be changed by the SMM service routine. The RSM instruction will update the SMM_BASE with the new value.
- The SMM_BASE is mapped to MSR C001_0111h and WRMSR instruction can be used to update it.

**SMM_BASE Address** **Offset FF00h**
**MSR C001_0111h**




| Bit | Name | Function | R/W |
|---|---|---|---|
| 63–32 | reserved | RAZ | |
| 31–0 | SMM_BASE | System management mode base | R/W |

# 6.7 Auto Halt Restart

During entry into SMM, the auto halt restart byte at offset FEC9h in the SMM state save area indicates whether SMM was entered from the Halt state. Before returning from SMM, the halt restart byte bit can be written by the SMM service routine to specify whether the return from SMM should take the processor back to the Halt state or to the instruction-execution state specified by the SMM state save area (the instruction after the HLT).

If the return from SMM takes the processor back to the Halt state, the HLT instruction is not refetched and reexecuted, but the Halt special bus cycle is sent to the system on the bus and the processor enters the HLT state.

### Auto Halt Restart — Offset FEC9h




| Bit | Name | Function | R/W |
|---|---|---|---|
| 7–1 | reserved | SBZ | R/W |
| 0 | HLT | HLT Restart Byte | R/W |

### Field Description

**HLT Restart Byte (HLT)**—Bit 0. On entry:

0 = Entered from normal x86 instruction boundary
1 = Entered from HLT State

After entry and before returning:

0 = Return to SMM State
1 = Return to HLT State

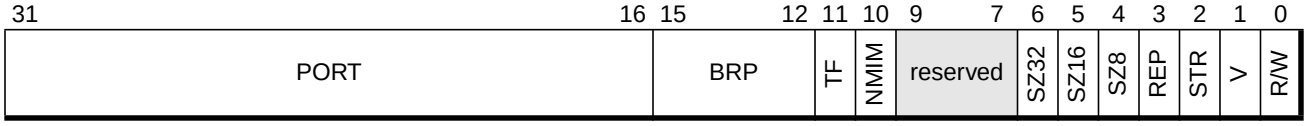
## 6.8 SMM I/O Trap and I/O Restart

If the assertion of SMI is recognized on the boundary of an I/O instruction, the I/O trap doubleword at offset FEC0h in the SMM state save area contains information about the executed I/O instruction. If the Valid bit is equal to 0, not all information is valid.

This is used in the I/O restart implementation. When an I/O access is done to a device that may be unavailable, the system can assert SMI and trap the I/O instruction. The system can then determine which device is not responding by using the I/O port information in the I/O Trap doubleword. It can then figure out why the device is not responding, fix the device, and re-execute the I/O instruction. It can reconstruct and then re-execute the I/O instruction in the SMM handler by using the SMM I/O trap information. More likely, it can use the SMM I/O restart mechanism to cause the processor to restart the I/O instruction immediately after the RSM.

## 6.8.1 SMM I/O Trap

**SMM I/O Trap** **Offset FEC0h**




| Bit | Name | Function | R/W |
|---|---|---|---|
| 31–16 | PORT | Trapped I/O Port | R |
| 15–12 | BRP | I/O Breakpoint Matches | R |
| 11 | TF | EFLAGS TF Value | R |
| 10 | NMIM | NMI Mask | R |
| 9–7 | reserved | RAZ | R |
| 6 | SZ32 | Port Access was 32-bit | R |
| 5 | SZ16 | Port Access was 16-bit | R |
| 4 | SZ8 | Port Access was 8-bit | R |
| 3 | REP | Repeated Port Access | R |
| 2 | STR | String Based Port Access | R |
| 1 | V | I/O Trap Word Valid Bit | R |
| 0 | R/W | Port Access Type<br>0 = Write (OUT instruction)<br>1 = Read (IN instruction) | R |

## 6.8.2 SMM I/O Restart Byte

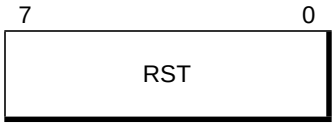
The processor initializes the I/O trap restart slot to 00h on entry into SMM. If SMM is entered as a result of a trapped I/O instruction, the processor indicates the validity of the I/O instruction by setting or clearing bit 1 of the I/O trap doubleword at offset FEC0h in the SMM state save area. The SMM service routine should test bit 1 of the I/O trap doubleword to determine if a valid I/O instruction was being executed when entering SMM and before writing the I/O trap restart slot. If the I/O instruction is valid, the SMM service routine can safely rewrite the I/O trap restart slot with the value FFh, causing the processor to re-execute the trapped I/O instruction when the RSM instruction is executed. If the I/O instruction is invalid, writing the I/O trap restart slot has undefined results.

If a second SMI# is asserted and a valid I/O instruction is trapped by the first SMM handler, the CPU services the second SMI# prior to re-executing the trapped I/O instruction. The second entry into SMM will not have bit 1 of the I/O trap doubleword set, and the second SMM service routine must not rewrite the I/O trap restart slot.

During a simultaneous SMI# I/O instruction trap and debug breakpoint trap, the processor first responds to the SMI# and postpones recognizing the debug exception until after the RSM instruction returns from SMM. If the debug registers DR3–DR0 are used while in SMM, they must be saved and restored by the SMM handler. The processor automatically saves and restores DR7 and DR6. If the I/

O trap restart slot in the SMM state save area contains the value FFh when the RSM instruction is executed, the debug trap does not occur until after the I/O instruction is re-executed.

**SMM I/O Restart Byte** **Offset FEC8h**




| Bit | Name | Function | R/W |
|---|---|---|---|
| 7–0 | RST | I/O Restart Byte<br>00h = Do not restart<br>FFh = Restart I/O instruction | R/W |

## 6.9 Exceptions and Interrupts in SMM

When SMM is entered, the processor masks INTR, NMI, SMI, INIT, and A20M interrupts. The processor clears the IF flag to disable INTR interrupts. To enable INTR interrupts within SMM, the SMM handler must set the IF flag to 1. A20M is disabled so that address bit 20 is always generated externally when in SMM.

Generating an INTR interrupt is a method for unmasking NMI interrupts in SMM. The processor recognizes the assertion of NMI within SMM immediately after the completion of an IRET instruction. The NMI can be enabled by using an INTR interrupt. Once NMI is recognized within SMM, NMI recognition remains enabled until SMM is exited, at which point NMI masking is restored to the state it was in before entering SMM.

Because the IF flag is cleared when entering SMM, the HLT instruction should not be executed in SMM without first setting the IF bit to 1. Setting this bit to 1 enables the processor to exit the Halt state by means of an INTR interrupt.

The processor still responds to the DBREQ and STPCLK interrupts as well as to all exceptions that might be caused by the SMM handler.

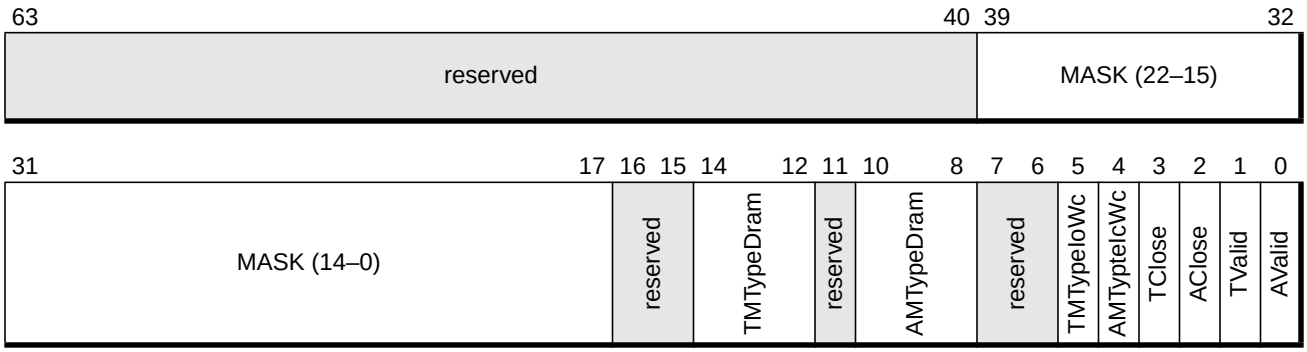
## 6.10 Protected SMM and ASeg/TSeg

System management memory, SMRAM, is defined by two ranges. The ASeg range is located at a fixed range from A0000h–BFFFFh. The TSeg ranges it located at a variable base. The size of the TSeg range is controlled by a variable mask. SMRAM provides a safe location for system management code and data that is not readily accessible by applications. The SMM interrupt handler can be located in one of these two ranges if protection is needed, or it can be located outside these ranges in the rest of memory.

### 6.10.1 SMM_MASK Register

This register holds the mask of the TSeg range as well as configuration information for both the fixed ASeg range and the variable TSeg range.

**SMM_MASK Register** **MSR C001_0113h**




| Bit | Name | Function | R/W |
|---|---|---|---|
| 63–40 | reserved | RAZ | |
| 39–17 | MASK | SMRAM TSeg Range Mask | R/W |
| 16–15 | reserved | RAZ | |
| 14–12 | TMTypeDram | SMRAM TSeg Range Memory Type | R/W |
| 11 | reserved | RAZ | |
| 10–8 | AMTypeDram | SMRAM ASeg Range Memory Type | R/W |
| 7–6 | reserved | RAZ | |
| 5 | TMTypeIoWc | Non-SMRAM TSeg Range Memory Type | R/W |
| 4 | AMTypteIcWc | Non-SMRAM ASeg Range Memory Type | R/W |
| 3 | TClose | Send TSeg Range Data Accesses to Non-SMRAM | R/W |
| 2 | AClose | Send ASeg Range Data Accesses to Non-SMRAM | R/W |
| 1 | TValid | Enable TSeg SMRAM Range | R/W |
| 0 | AValid | Enable ASeg SMRAM Range | R/W |

**SMRAM TSeg Range Mask (MASK)**—Bits 39–17. Mask of the SMRAM TSeg Range.

**SMRAM TSeg Range Memory Type (TMTypeDram)**—Bits 14–12. Memory Type for the SMRAM TSeg Range.

**SMRAM ASeg Range Memory Type (AMTypeDram)**—Bits 10–8. Memory Type for the SMRAM ASeg Range.

**Non-SMRAM TSeg Range Memory Type (TMTypeIoWc)**—Bit 5. Memory Type for the non-SMRAM TSeg Range.

**Non-SMRAM ASeg Range Memory Type (AMTypteIcWc)**—Bit 4. Memory Type for the non-SMRAM ASeg Range.

**Send TSeg Range Data Accesses to Non-SMRAM (TClose)**—Bit 3. Send Data Accesses to the TSeg Range to Non-SMRAM.

**Send ASeg Range Data Accesses to Non-SMRAM (AClose)**—Bit 2. Send Data Accesses to the ASeg Range to Non-SMRAM.

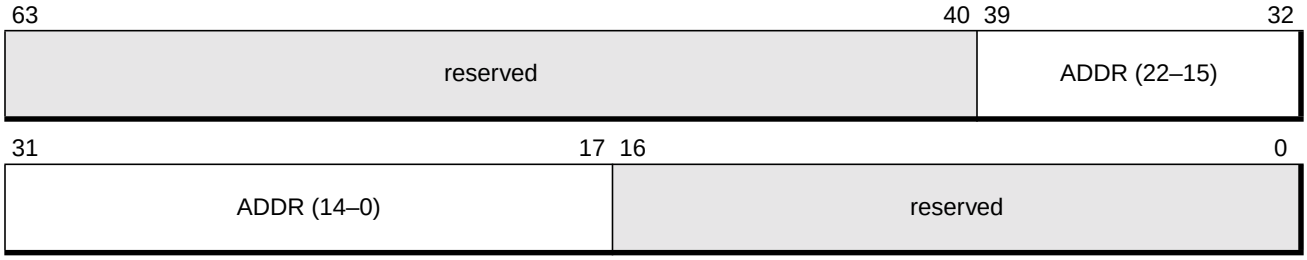
**Enable TSeg SMRAM Range (TValid)**—Bit 1. Enables the TSeg SMRAM Range.

**Enable ASeg SMRAM Range (AValid)**—Bit 0. Enables the ASeg SMRAM Range.

## 6.10.2 SMM_ADDR Register

This register holds the base of the variable TSeg range.

**SMM_ADDR Register** **MSR C001_0112h**




| Bit | Name | Function | R/W |
|---|---|---|---|
| 63–40 | reserved | RAZ | |
| 39–17 | ADDR | Base of the SMRAM TSeg Range | R/W |
| 16–0 | reserved | RAZ | |

## 6.10.3 SMM ASeg

The SMM ASeg address range is A0000–BFFFFh. The ASeg is enabled by the AValid bit in the SMM_MASK MSR.

When the ASeg is enabled, the MTRR memory maps are not used for the addresses in A0000–BFFFFh memory range. Instead, settings in the SMM_MASK register and whether the processor is in SMM or not determine how those addresses are handled. If not in SMM, the addresses are mapped to I/O space and do not access DRAM. The memory type used is either UC or WC, based on the AMTypeIoWc bit in the SMM_MASK register. When inside SMM, the accesses to the ASeg memory range are directed to DRAM.

The memory type is determined using the normal memory type encoding and is specified in AMTypeDram. Table 36 lists these types.

**Table 36. SMM ASeg-Enabled Memory Types**

| Address Range | In SMM | Out of AMTypeIoWc=0 | SMM AMTypeIoWc=1 |
|---|---|---|---|
| 0–9FFFh | Normal MTRR | Normal MTRR | Normal MTRR |
| A0000–BFFFFh | DRAM, AMTypeDram | I/O, UC | I/O, WC |
| C0000h–Max | Normal MTRR | Normal MTRR | Normal MTRR |

The SMM save state and code can be cached. After leaving SMM, the same addresses will bypass the caches and DRAM and access the I/O memory subsystem. This protects the SMM memory from corruption by programs outside of SMM.

## 6.10.4 SMM TSeg

The TSeg is similar to the ASeg except it provides more programability and therefore allows more space to be allocated in a different area of memory for the SMM handler. It is enabled by the TValid bit in the SMM_MASK register. It uses the SMM_ADDR register to specify its base. It can be used with ASeg or by itself. When it is used with ASeg, the two spaces should not overlap. If they do overlap, the memory types should be consistent in both the ASeg map and the TSeg map for the overlapping addresses. Otherwise, undefined behavior will occur.

The SMM TSeg begins at the address specified in SMM_ADDR concatenated with 0s in the lower 17 bits, forcing the TSeg to begin on a 128-Kbyte boundary. The ending of TSeg is specified by the SMM_Mask[39:17]. The SMM_Mask and SMM_ADDR function as the variable-range MTRRs. Each address generated by the processor is ANDed with the SMM_MASK and compared to the SMM_ADDR ANDed with the mask. If the values are equal, the address is within the TSeg range. For example, suppose you wish to create a TSeg starting at the 1-Mbyte address boundary and extending 256 Kbytes. The SMM_ADDR register would be set to 0010_0000h and the SMM_MASK to FFFC_0002h. This will make the TSeg range from 0010_0000–0013_FFFFh.

The TSeg memory type table is similar to the ASeg memory type table. When not in SMM and TSeg is enabled, the addresses within the TSeg range are directed to I/O space with either a UC memory type or a WC memory type, based on the TMTypeIoWc bit in the SMM_MASK register. When within SMM, the accesses are directed to DRAM with a memory type specified by TMTypeDram.

**Table 37. SMM TSeg-Enabled Memory Types**

| Address Range | In SMM | Out of SMM | |
|---|---|---|---|
| | | TMTypeIoWc=0 | TMTypeIoWc=1 |
| <TSeg Range | Normal MTRR | Normal MTRR | Normal MTRR |
| TSeg Range | DRAM, TMTypeDram | I/O, UC | I/O, WC |
| > TSeg Range | Normal MTRR | Normal MTRR | Normal MTRR |

### 6.10.5 Closing SMM

Sometimes within SMM code with ASeg or TSeg enabled, there is a requirement to access the I/O space at the same address as the current SMM segment. That is typically only accessible outside of SMM. To accomplish this function, the Aclose and Tclose bits from SMM_MASK register are used. When the Aclose bit is set, data cache accesses to the ASeg that would normally go to DRAM are redirected to I/O, with the memory type specified by AMTypeIoWc.

The same function applies to the TSeg. Instruction cache accesses and Page Directory/Table accesses still access the SMM code in DRAM. When the SMM handler is done accessing the I/O space, it must clear the appropriate close bit. Failure to do so and then issuing an RSM will probably cause the processor to enter shutdown, as the save state will be read from I/O space.

### 6.10.6 Locking SMM

The SMM registers can be locked by setting the SMMLOCK (HWCR, bit 0). Once set, the SMM_BASE, the SMM_ADDR, all but the two close bits of SMM_MASK and the RSMSPCYCDIS, SMISPCYCDIS, and SMMLOCK bits of HWCR are locked and cannot be changed. The only way to unlock the SMM registers is to assert reset. This provides security to the SMM mechanism. The BIOS can lock the SMM environment after setting it up so that it can not be tampered with.

## 6.11 SMM Special Cycles

Special cycles can be initiated on entry and exit from SMM to acknowledge to the system that these transitions are occurring. This allows secure spaces external to the processor to be secured within the SMM environment as well. It also helps with synchronizing the SMI interrupt across multiple processor systems. These special cycles can be disabled by setting the SMISPCYCDIS to 1 to disable the entry special cycle and setting the RSMSPCYCDIS bit to 1 to disable the exit special cycle.

# 7 Advanced Programmable Interrupt Controller (APIC)

All AMD Athlon™ 64 and AMD Opteron™ processor-based systems must support 64-bit operating systems. As a result, all platforms must support APIC and declare it to the operating system in the appropriate tables.

The local APIC contains logic to receive interrupts from a variety of sources and to send interrupts to other local APICs, as well as registers to control its behavior and report status.

Interrupts can be received from:

- HyperTransport™ I/O devices, including the I/O hub (I/O APICs)
- Other local APICs (inter-processor interrupts)
- APIC timer
- Performance counters
- Legacy local interrupts from the I/O hub (INTR and NMI)
- APIC internal errors

The APIC timer, performance counters, local interrupts, and internal errors are all considered local interrupt sources, and their routing is controlled by local vector table entries. These entries assign a message type and vector to each interrupt, allow them to be masked, and track the status of the interrupt.

I/O and inter-processor interrupts have their message type and vector assigned at the source and are unaltered by the local APIC. They carry a destination field and a mode bit that together determine which local APICs will accept them. The destination mode (DM) bit specifies if the interrupt request packet should be handled in Physical or Logical destination mode.

- In physical destination mode, if the destination field is FFh, the interrupt is a broadcast and is accepted by all local APICs. Otherwise, the interrupt is only accepted by the local APIC whose APIC ID matches the destination field of the interrupt. Physical mode allows up to 255 APICs to be addressed individually.

- In logical destination mode, a local APIC accepts interrupts selected by the logical destination register (LOG_DEST) and the destination field of the interrupt using either cluster or flat format, as configured by the destination format register.
  - If flat destinations are in use, bits 7–0 of the LOG_DEST field are checked against bits 7–0 of the arriving interrupt's Destination Field. If any bit position is set in both fields, this APIC is a valid destination. Flat format allows up to 8 APICs to be addressed individually.
  - If cluster destinations are in use, bits 7–4 of the LOG_DEST field are checked against bits 7–4 of the arriving interrupt's destination field, identifying the cluster. If all of bits 7–4 match, then bits 3–0 of the LOG_DEST and the interrupt destination are checked for any bit positions that are set in both fields, choosing processors within the cluster. If both conditions are met, this APIC is a valid destination. Cluster format allows 15 clusters of 4 APICs each to be addressed.
  - In both flat and cluster formats, if the destination field is FFh, the interrupt is a broadcast and is accepted by all local APICs.

## 7.1 Interrupt Delivery

SMI, NMI, INIT, Startup, and External interrupts are classified as non-vectored interrupts.

When an APIC accepts a non-vectored interrupt, it is handled directly by the processor instead of being queued in the APIC. When an APIC accepts a fixed or lowest priority interrupt, it sets the bit in the interrupt request register (IRR) corresponding to the vector in the interrupt. (For local interrupt sources, this comes from the vector field in that interrupt's local vector table entry.) If a subsequent interrupt with the same vector arrives when the corresponding IRR bit is already set, the two interrupts are collapsed into one. Vectors 0–15 are reserved.

## 7.2 Vectored Interrupt Handling

The task priority register (TPR_PRI) and processor priority register (PROC_PRI) each contain an 8-bit priority, divided into a main priority (bits 7–4) and a priority sub-class (3–0). The task priority is assigned by software to set a threshold priority at which the processor will be interrupted.

To calculate processor priority, an 8-bit ISR vector is set based on the highest bit set in the in-service register (ISR). Like the TPR_PRI and PROC_PRI, this vector is divided into a main priority (7–4) and priority sub-class (3–0). The main priority of the TPR_PRI and ISR are compared and the highest is the PROC_PRI, as follows:

```
If (TPR_PRI[7:4] >= ISRVect[7:4])PROC_PRI = TPR_PRI
Else                             PROC_PRI = {ISRVect[7:4],0h}
```

The resulting processor priority is used to determine if any accepted interrupts (indicated by IRR bits) are high enough priority to be serviced by the processor. When the processor is ready to service an interrupt, the highest bit in the IRR is cleared, and the corresponding bit is set in the ISR. The

corresponding trigger mode register (TMR) bit is set if the interrupt is level-triggered and cleared if edge-triggered.

When the processor has completed service for an interrupt, it performs a write to the end of interrupt (EOI) register, clearing the highest ISR bit and causing the next-highest interrupt to be serviced. If the corresponding TMR bit is set, EOI messages will be sent to all APICs to complete service of the interrupt at the source.

## 7.3 Spurious Interrupts

In the event that the task priority is set to or above the level of the interrupt to be serviced, the local APIC will deliver a spurious interrupt vector to the processor, as specified by the spurious interrupt vector register. The ISR will be unchanged and no EOI will occur.

### 7.3.1 Spurious Interrupts Caused by Timer Tick Interrupt

A typical interrupt is asserted until it is serviced. Interrupt is deasserted when software clears the interrupt status bit within the interrupt service routine. Timer tick interrupt is an exception, since it is deasserted regardless of whether it is serviced or not. A BIOS programs the 8254 Programmable Timer to generate an 18.2 Hz square wave. This square wave represents the timer tick interrupt (PITIRQ) and in some interrupt configurations it is connected to the 8259 Programmable Interrupt Controller (PIC) IRQ0 input.

The processor is not always able to service interrupts immediately (i.e. when interrupts are masked by clearing EFLAGS.IM or when the processor is in debug mode). The method of interrupt delivery to the processor (wire or messages) determines system behavior for the timer tick interrupt connected to the PIC after the processor has not been able to service it for some time. The PIC output INTR is identical to PITIRQ when interrupts are not serviced.

If interrupts are delivered to the processor using a wire, and the processor is not able to service the timer tick interrupt for some time, timer tick interrupts asserted during that time will be lost. The following cases are possible when the processor is ready to service interrupts:

- INTR is deasserted and it is not serviced by the processor. This will happen almost 50 percent of the time.
- INTR is asserted and it is serviced by the processor. This will happen almost 50 percent of the time.
- INTR is asserted, and it is detected by the processor. The processor sends the interrupt acknowledge cycle, but when the PIC receives it, INTR is deasserted, and the PIC sends a spurious interrupt vector. This will happen a very small percentage of the time.

The probability of spurious interrupts, when interrupts are delivered to the processor using a wire, is very low. An example of a configuration that uses wire to deliver interrupts to the processor has PITIRQ connected to PIC IRQ0, and PIC INTR connected to LINT0 of an AMD Athlon processor.

If interrupts are delivered to the processor using messages, and the processor is not able to service the timer tick interrupt for an extended period of time, the INTR caused by the first timer tick interrupt asserted during that time is delivered to the local APIC in ExtInt mode and latched, and the subsequent timer tick interrupts are lost. The following cases are possible when the processor is ready to service interrupts:

- An ExtInt interrupt is pending, and INTR is asserted. This results in timer tick interrupt servicing. This occurs 50 percent of the time.
- An ExtInt interrupt is pending, and INTR is deasserted. The processor sends the interrupt acknowledge cycle, but when the PIC receives it, INTR is deasserted, and the PIC sends a spurious interrupt vector. This occurs 50 percent of the time.

There is a 50 percent probability of spurious interrupts when interrupts are delivered to the processor using messages. An example of an AMD Athlon™ 64 or AMD Opteron™ processor configuration that uses messages to deliver interrupts to the processor has PITIRQ connected to PIC IRQ0, PIC INTR connected to IOAPIC IRQ0, and interrupts are delivered to the processor by HyperTransport messages. HyperTransport messages are always used to deliver PIC interrupts to the processor in a system with AMD Athlon™ 64 or AMD Opteron™ processors. An example of an AMD Athlon processor configuration that uses messages to deliver interrupts to the processor has PITIRQ connected to PIC IRQ0, PIC INTR connected to IOAPIC IRQ0, and interrupts are delivered to the processor by APIC 3-wire messages.

## 7.4 Lowest-Priority Arbitration

Fixed and non-vectored interrupts are accepted by their destination APICs without arbitration.

Delivery of lowest-priority interrupts requires all APICs to arbitrate to determine which one will accept the interrupt. If focus processor checking is enabled (bit 9 of the spurious interrupt vector register cleared), then the focus processor for an interrupt will always accept the interrupt. A processor is the focus of an interrupt if it is already servicing that interrupt (corresponding ISR bit is set) or if it already has a pending request for that interrupt (corresponding IRR bit is set). If there is no focus processor for an interrupt, or focus processor checking is disabled, then each APIC will calculate an arbitration priority value, and the one with the lowest result will accept the interrupt.

To calculate arbitration priority, an 8-bit IRR Vector is set based on the highest bit set in the IRR. Like the ISR, TPR_PRI, and PROC_PRI, this vector is divided into a main priority (7–4) and priority sub-class (3–0). The main priority of the TPR_PRI, ISR, and IRR are compared and the highest is the new ARB_PRI, as follows:

```
If   (TPR_PRI[7:4] >= IRRVect[7:4] and
      TPR_PRI[7:4] >  ISRVect[7:4])    ARB_PRI = TPR_PRI
Else if (IRRVect[7:4] > ISRVect[7:4]) ARB_PRI = {IRRVect[7:4],0h}
Else                                  ARB_PRI = {ISRVect[7:4],0h}
```

## 7.5 Inter-Processor Interrupts

In order to redirect an interrupt to another processor, originate an interrupt to another processor, or allow a processor to interrupt itself, the interrupt command register (ICR) provides a mechanism for generating interrupts. A write to the low doubleword of the ICR causes an interrupt to be generated, with the properties specified by the ICR low and ICR high fields.

## 7.6 APIC Timer Operation

The local APIC contains a 32-bit timer, controlled by the Timer LVT entry, initial count, and divide configuration registers. The processor bus clock is divided by the value in the timer divide configuration register to obtain a time base for the timer. When the timer initial count register is written, the value is copied into the timer current count register. The current count register is decremented at the rate of the divided clock. When the count reaches 0, a timer interrupt is generated with the vector specified in the Timer LVT entry. If the Timer LVT entry specifies periodic operation, the current count register is reloaded with the initial count value, and it continues to decrement at the rate of the divided clock. If the mask bit in the timer LVT entry is set, timer interrupts are not generated.

## 7.7 State at Reset

At power-up or reset, all registers take on the values listed in their descriptions. SMI, NMI, INIT, Startup, and LINT interrupts may be accepted.

When the APIC is software-disabled, pending interrupts in the ISR and IRR are held, but further fixed, lowest-priority, and ExtInt interrupts will not be accepted. All LVT entry mask bits are set and cannot be cleared.

When a processor accepts an INIT interrupt, the APIC is reset as at power-up, with the exception that APIC_ID is unaffected.

# 7.8 Register Summary

All APIC registers are accessed with memory reads and writes of (APIC_BASE+Offset). APIC_BASE is MSR 001Bh and defaults to 00_FEE0_0000h.

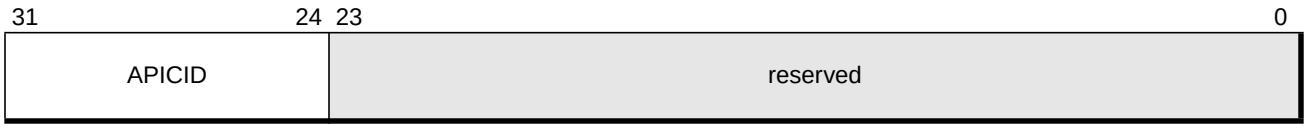
The APIC registers and their offsets are listed in Table 38.

**Table 38. APIC Register Summary**

| Offset (Hex) | Mnemonic | Name |
|---|---|---|
| 20 | APIC_ID | APIC ID Register |
| 30 | APIC_VER | APIC Version Register |
| 80 | TPR_PRI | Task Priority Register |
| 90 | ARB_PRI | Arbitration Priority Register |
| A0 | PROC_PRI | Processor Priority Register |
| B0 | EOI | End Of Interrupt Register |
| D0 | LOG_DEST | Logical Destination Register |
| E0 | DEST_FORMAT | Destination Format Register |
| F0 | SPUR_INTR_VEC | Spurious Interrupt Vector Register |
| 100-170 | ISR | In-Service Registers |
| 180-1F0 | TMR | Trigger Mode Registers |
| 200-270 | IRR | Interrupt Request Registers |
| 280 | ERR_STAT | Error Status Register |
| 300 | ICRLO | Interrupt Command Register Low (bits 31–0) |
| 310 | ICRHI | Interrupt Command Register High (bits 63–32) |
| 320 | TIMER_LVT | Timer Local Vector Table Entry |
| 340 | PERF_CNT_LVT | Performance Counter Local Vector Table Entry |
| 350 | LINT0_LVT | Local Interrupt 0 Local Vector Table Entry |
| 360 | LINT1_LVT | Local Interrupt 1 Local Vector Table Entry |
| 370 | ERROR_LVT | Error Local Vector Table Entry |
| 380 | INIT_CNT | Timer Initial Count Register |
| 390 | CURR_CNT | Timer Current Count Register |
| 3E0 | TIMER_DVD_CFG | Timer Divide Configuration Register |

## 7.8.1 APIC ID Register

### APIC_ID Register — Offset 20h



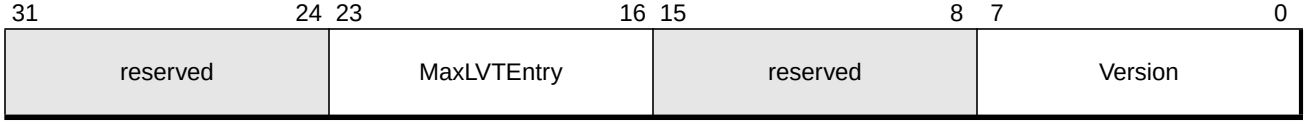

| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | APICID | APIC Identification | R/W | Node ID |
| 23–0 | reserved | | R/O | 00_0000h |

### Field Descriptions

**APIC Identification (APICID)**—Bits 31–24. Software must ensure that all APICs are assigned unique APIC IDs. When both ApicExtId and ApicExtBrdCst in the HyperTransport™ Transaction Control Register are set, all 8 bits of APIC ID are used. When either ApicExtID or ApicExtBrdCst is clear, only bits 3–0 of APIC ID are used, and bits 7–4 are reserved. Node ID is initially 00h if this is the boot strap processor or 07h for all other nodes.

## 7.8.2 APIC Version Register

### APIC_VER Register — Offset 30h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | reserved | | R/O | 00h |
| 23–16 | MaxLVTEntry | Maximum LVT Entry | R/O | 04h |
| 15–8 | reserved | | R/O | 00h |
| 7–0 | Version | Version | R/O | 10h |

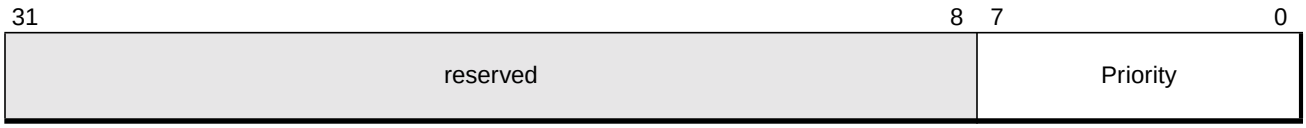
### Field Descriptions

**Max LVT Entry**—Bits 23–16. This field indicates the number of entries in the Local Vector Table minus one.

**Version**—Bits 7–0. This field indicates the version number of this APIC implementation.

## 7.8.3 Task Priority Register

### TPR_PRI Register — Offset 80h



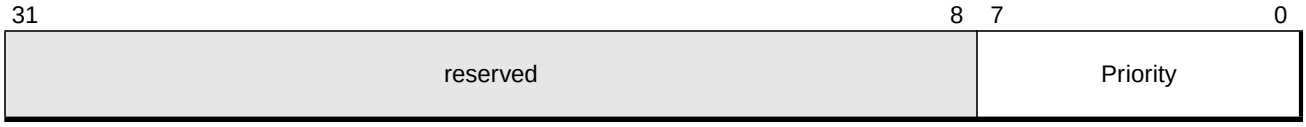

| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | R/O | 0000_00h |
| 7–0 | Priority | Priority | R/W | 00h |

### Field Descriptions

**Priority**—Bits 7–0. This field is assigned by software to set a threshold priority at which the processor will be interrupted.

## 7.8.4 Arbitration Priority Register

### ARB_PRI Register — Offset 90h



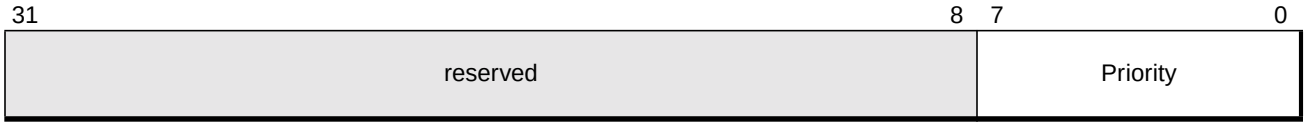

| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | R/O | 0000_00h |
| 7–0 | Priority | Priority | R/O | 00h |

### Field Descriptions

**Priority**—Bits 7–0. This field indicates the processor's current priority, for a task being serviced, an interrupt being serviced, or an interrupt that is pending, and is used to arbitrate between processors to determine which will accept a lowest-priority interrupt request.

## 7.8.5 Processor Priority Register

### PROC_PRI Register — Offset A0h



| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | R/O | 0000_00h |
| 7–0 | Priority | Priority | R/O | 00h |

### Field Descriptions

**Priority**—Bits 7–0. This field indicates the processor's current priority servicing a task or interrupt, and is used to determine if any pending interrupts should be serviced.

## 7.8.6 End of Interrupt Register

This register is written by the software interrupt handler to indicate that servicing of the current interrupt is complete.

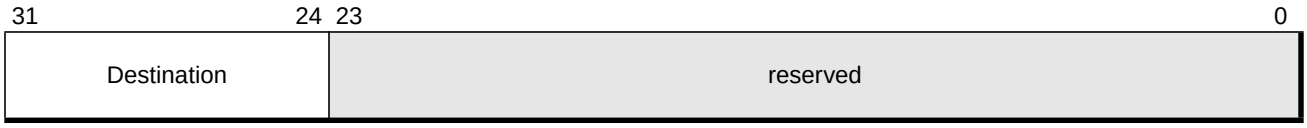
### EOI Register — Offset B0h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–0 | reserved | | W/O | XXXX_XXXXh |

## 7.8.7 Logical Destination Register

### LOG_DEST Register — Offset D0h



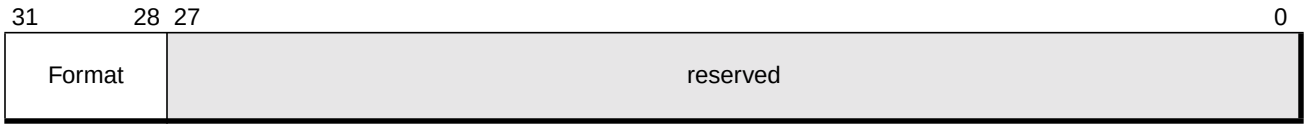

| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–24 | Destination | Destination | R/W | 00h |
| 23–0 | reserved | | R/O | 00_0000h |

### Field Descriptions

**Destination (Destination)**—Bits 31–24. This field contains this APIC's destination identification, and is used to determine which interrupts should be accepted.

## 7.8.8 Destination Format Register

### DEST_FORMAT Register — Offset E0h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–28 | Format | Format | R/W | Fh |
| 27–0 | reserved | | R/O | FFF_FFFFh |

### Field Descriptions

**Format (Format)**—Bits 31–28. 0h and Fh are the only allowed values. This field controls which format to use when accepting interrupts with a logical destination mode.
0h = Cluster destinations are used.
Fh = Flat destinations are used.

## 7.8.9 Spurious Interrupt Vector Register

### SPUR_INTR_VEC Register — Offset F0h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–10 | reserved | | R/O | 0000_00h |
| 9 | FocusDisable | Focus Disable | R/W | 0 |
| 8 | APICSWEn | APIC Software Enable | R/W | 0 |
| 7–0 | Vector | Vector | R/W | FFh |

### Field Descriptions

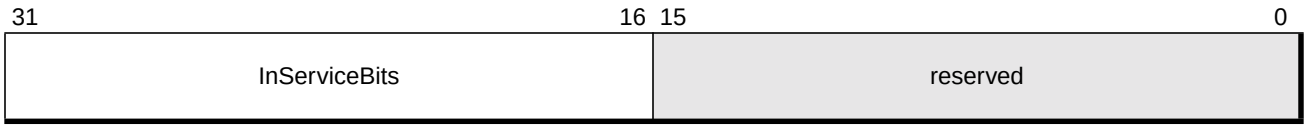
**Focus Disable (FocusDisable)**—Bit 9. This bit disables focus processor checking during lowest-priority arbitrated interrupts.

**APIC Software Enable (APICSWEn)**—Bit 8. When APICSWEnable is cleared, SMI, NMI, INIT, Startup, and LINT interrupts may be accepted, pending interrupts in the ISR and IRR are held, but further Fixed, Lowest-Priority, and ExtInt interrupts will not be accepted. All LVT entry Mask bits are set and cannot be cleared.

**Vector**—Bits 7–0. When ApicExtSpur in the HyperTransport™ Transaction Control Register is set, bits 3–0 of Vector are writable. When ApicExtSpur is clear, bits 3–0 are read-only 1111b. This field contains the vector that is sent to the processor in the event of a spurious interrupt.

## 7.8.10 In-Service Registers

### ISR Register — Offset 100h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | InServiceBits | In-Service Bits | R/O | 0000h |
| 15–0 | reserved | | R/O | 0000h |

### Field Descriptions

**In-Service Bits (InServiceBits)**—Bits 31–16. These bits are set when the corresponding interrupt is being serviced by the CPU.

### ISR Registers — Offsets 170h, 160h, 150h, 140h, 130h, 120h, 110h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–0 | InServiceBits | In-Service Bits | R/O | 0000_0000h |

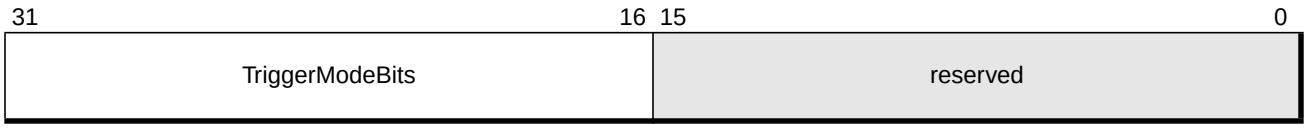
### Field Descriptions

**In-Service Bits (InServiceBits)**—Bits 31–0. These bits are set when the corresponding interrupt is being serviced by the CPU. Interrupts are mapped as follows:

| ISR | Interrupt number |
|---|---|
| Offset 110h | 63–32 |
| Offset 120h | 95–64 |
| Offset 130h | 127–96 |
| Offset 140h | 159–128 |
| Offset 150h | 191–160 |
| Offset 160h | 223–192 |
| Offset 170h | 255–224 |

## 7.8.11 Trigger Mode Registers

### TMR Register — Offset 180h

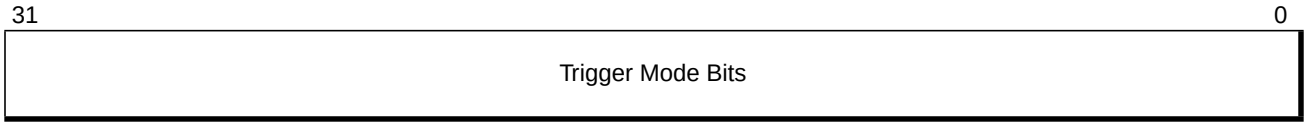



| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | TriggerModeBits | Trigger Mode Bits | R/O | 0000h |
| 15–0 | reserved | | R/O | 0000h |

### Field Descriptions

**Trigger Mode Bits (TriggerModeBits)**—Bits 31–16. The Trigger Mode bit for each corresponding interrupt is updated when an interrupt enters servicing. It is 0 for edge-triggered interrupts and 1 for level-triggered interrupts.

### TMR Registers — Offsets 1F0h, 1E0h, 1D0h, 1C0h, 1B0h, 1A0h, 190h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–0 | TriggerModeBits | Trigger Mode Bits | R/O | 0000_0000h |

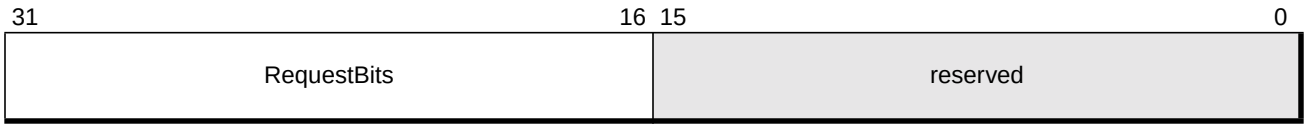
### Field Descriptions

**Trigger Mode Bits (TriggerModeBits)**—Bits 31–0. The Trigger Mode bit for each corresponding interrupt is updated when an interrupt enters servicing. It is 0 for edge-triggered interrupts and 1 for level-triggered interrupts. Interrupts are mapped as follows:

| TMR | Interrupt number |
|---|---|
| Offset 190h | 63–32 |
| Offset 1A0h | 95–64 |
| Offset 1B0h | 127–96 |
| Offset 1C0h | 159–128 |
| Offset 1D0h | 191–160 |
| Offset 1E0h | 223–192 |
| Offset 1F0h | 255–224 |

## 7.8.12 Interrupt Request Registers

### IRR Register — Offset 200h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–16 | RequestBits | Interrupt Requests Bits | R/O | 0000h |
| 15–0 | reserved | | R/O | 0000h |

### Field Descriptions

**Interrupt Request Bits (RequestBits)**—Bits 31–16. Request bits are set when the corresponding interrupt is accepted by the APIC.

### IRR Registers — Offsets 270h, 260h, 250h, 240h, 230h, 220h, 210h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–0 | RequestBits | Interrupt Requests Bits | R/O | 0000_0000h |

### Field Descriptions

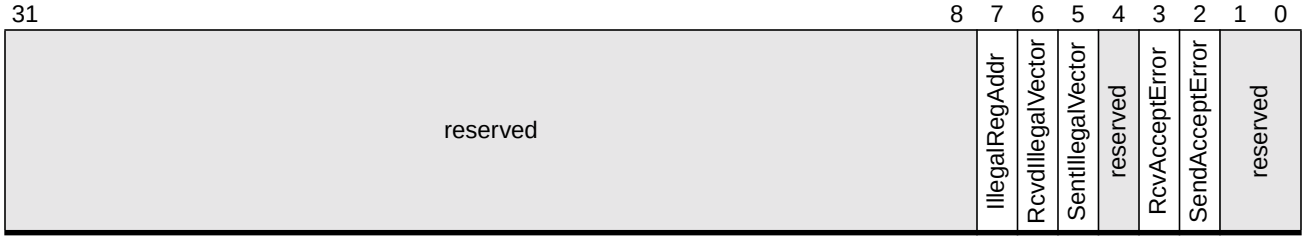
**Interrupt Request Bits (RequestBits)**—Bits 31–0. Request bits are set when the corresponding interrupt is accepted by the APIC. Interrupts are mapped as follows:

| IRR | Interrupt number |
|---|---|
| Offset 210h | 63–32 |
| Offset 220h | 95–64 |
| Offset 230h | 127–96 |
| Offset 240h | 159–128 |
| Offset 250h | 191–160 |
| Offset 260h | 223–192 |
| Offset 270h | 255–224 |

## 7.8.13 Error Status Register

This register must be written to trigger an update before it can be read. Each write causes the internal error state to be loaded into this register, clearing the internal error state. A second write before another error occurs causes this register to be cleared.

**ERR_STAT Register** **Offset 280h**




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–8 | reserved | | R/O | 0000_00h |
| 7 | IllegalRegAddr | Illegal Register Address | W/R | 0 |
| 6 | RcvdIllegalVector | Received Illegal Vector | W/R | 0 |
| 5 | SentIllegalVector | Sent Illegal Vector | W/R | 0 |
| 4 | reserved | | R/O | 0 |
| 3 | RcvAcceptError | Receive Accept Error | W/R | 0 |
| 2 | SendAcceptError | Send Accept Error | W/R | 0 |
| 1–0 | reserved | | R/O | 00b |

### Field Descriptions

**Illegal Register Address (IllegalRegAddr)**—Bit 7. This bit indicates that an access to a nonexistent register location within this APIC was attempted.

**Received Illegal Vector (RcvdIllegalVector)**—Bit 6. This bit indicates that this APIC received a message with an illegal Vector (00h to 0Fh for fixed and lowest priority interrupts).

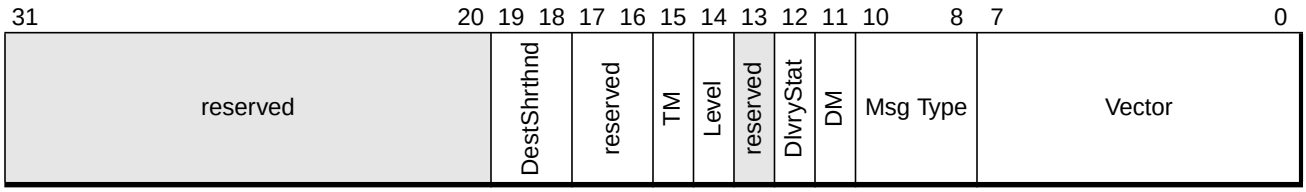
**Sent Illegal Vector (SentIllegalVector)**—Bit 5. This bit indicates that this APIC attempted to send a message with an illegal Vector (00h to 0Fh for fixed and lowest priority interrupts).

**Receive Accept Error (RcvAcceptError)**—Bit 3. This bit indicates that a message received by this APIC was not accepted by this or any other APIC.

**Send Accept Error (SendAcceptError)**—Bit 2. This bit indicates that a message sent by this APIC was not accepted by any APIC.

## 7.8.14 Interrupt Command Register Low

### ICRLO Register — Offset 300h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–20 | reserved | | R/O | 000h |
| 19–18 | DestShrthnd | Destination Shorthand | R/W | 00b |
| 17–16 | reserved | | R/O | 00b |
| 15 | TM | Trigger Mode | R/W | 0 |
| 14 | Level | Level | R/W | 0 |
| 13 | reserved | | R/O | 0 |
| 12 | DlvryStat | Delivery Status | R/O | 0 |
| 11 | DM | Destination Mode | R/W | 0 |
| 10–8 | MsgType | Message Type | R/W | 000b |
| 7–0 | Vector | Vector | R/W | 00h |

### Field Descriptions

**Destination Shorthand (DestShrthnd)**—Bits 19–18. This field provides a quick way to specify a destination for a message. If All Including Self or All Excluding Self are used, then DM is ignored and physical is automatically used.

00b =Destination Field
01b =Self
10b =All Including Self
11b =All Excluding Self (Note that this sends a message with a destination encoding of all 1's, so if lowest priority is used, the message could end up being reflected back to this APIC.)

**Trigger Mode (TM)**—Bit 15. This bit can be 0 for Edge or 1 for Level.

**Level (Level)**—Bit 14. This bit can be 0 for deasserted or 1 or asserted.

**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the destination CPU(s).

**Destination Mode (DM)**—Bit 11. This bit can be 0 for Physical or 1 for Logical.**Message Type (MsgType)**—Bits 10–8. The encoding for this field is as follows:

000b = Fixed
001b = Lowest Priority
010b = SMI
011b = Remote Read

100b = NMI
101b = INIT
110b = Startup
111b = External Interrupt

***Note:*** *Not all combinations of ICR fields are valid. Only those combinations listed in Table 39 are valid.*

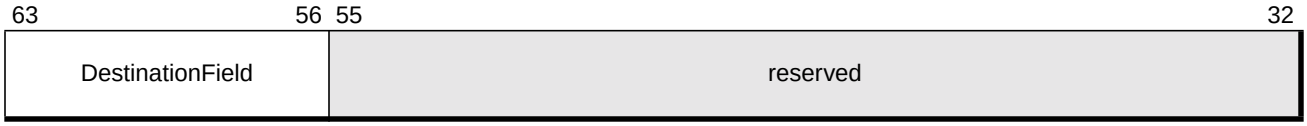
**Table 39. Valid Combinations of ICR Fields**

| Message Type (MsgType) | Trigger Mode (TM) | Level (Lvl) | Destination Shorthand (DestShrthnd) |
|---|---|---|---|
| Fixed | Edge | X | X |
| | Level | Assert | X |
| Lowest Priority, SMI, NMI, INIT | Edge | X | Dest or All Excluding Self |
| | Level | Assert | Dest or All Excluding Self |
| Startup | X | X | Dest or All Excluding Self |
| ***Note:*** *X indicates a "don't care".* | | | |

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 7.8.15 Interrupt Command Register High

### ICRHI Register — Offset 310h



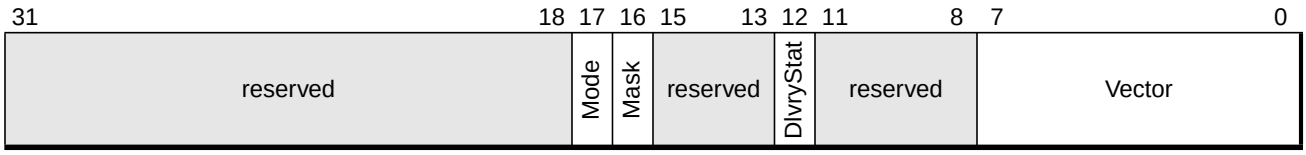

| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | DestinationField | Destination Field | R/W | 00h |
| 55–32 | reserved | | R/O | 00_0000h |

### Field Descriptions

**Destination Field (DestinationField)**—Bits 63–56. This field contains the destination encoding used when the destination shorthand is 00b.

## 7.8.16 Timer Local Vector Table Entry

### TIMER_LVT Register — Offset 320h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–18 | reserved | | R/O | 0000h |
| 17 | Mode | Mode | R/W | 0 |
| 16 | Mask | Mask | R/W | 1 |
| 15–13 | reserved | | R/O | 000b |
| 12 | DlvryStat | Delivery Status | R/O | 0 |
| 11–8 | reserved | | R/O | 0h |
| 7–0 | Vector | Vector | R/W | 00h |

### Field Descriptions

**Mode (Mode)**—Bit 17. This bit is 0 for One-shot and 1 for Periodic.

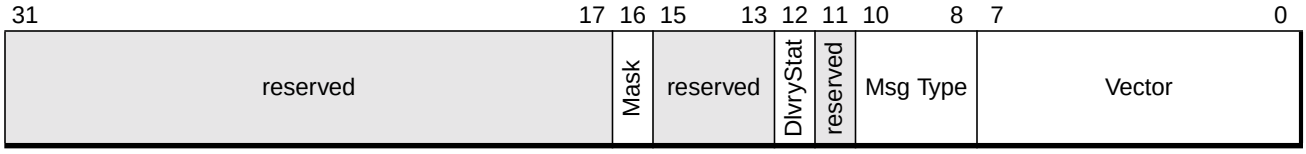
**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 7.8.17 Performance Counter Local Vector Table Entry

### PERF_CNT_LVT Register — Offset 340h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–17 | reserved | | R/O | 0000h |
| 16 | Mask | Mask | R/W | 1 |
| 15–13 | reserved | | R/O | 000b |
| 12 | DlvryStat | Delivery Status | R/O | 0 |
| 11 | reserved | | R/O | 0 |
| 10–8 | MsgType | Message Type | R/W | 000b |
| 7–0 | Vector | Vector | R/W | 00h |

### Field Descriptions

**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

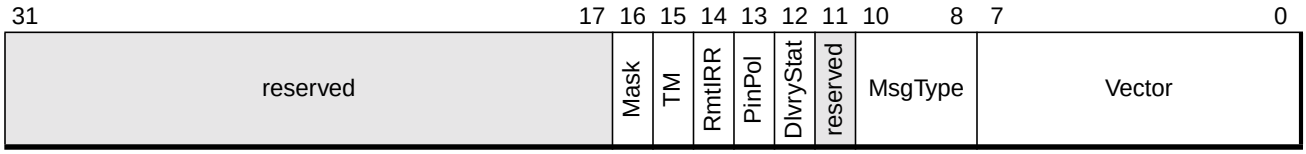
**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

**Message Type (MsgType)**—Bits 10–8. Only Message Types 000b (Fixed) and 100b (NMI) are legal.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 7.8.18 Local Interrupt 0 (Legacy INTR) Local Vector Table Entry Register

**LINT0_LVT Register** **Offset 350h**




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–17 | reserved | | R/O | 0000h |
| 16 | Mask | Mask | R/W | 1 |
| 15 | TM | Trigger Mode | R/W | 0 |
| 14 | RmtIRR | Remote IRR | R/O | 0 |
| 13 | PinPol | Pin Polarity | R/W | 0 |
| 12 | DlvryStat | Delivery Status | R/O | 0 |
| 11 | reserved | | R/O | 0 |
| 10–8 | MsgType | Message Type | R/W | 000b |
| 7–0 | Vector | Vector | R/W | 00h |

### Field Descriptions

**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

**Trigger Mode (TM)**—Bit 15. This bit is set for level-triggered interrupts and clear for edge-triggered interrupts.

**Remote IRR (RmtIRR)**—Bit 14. If trigger mode is level, remote IRR is set when the interrupt has begun service. Remote IRR is cleared when the EOI has occurred.

**Pin Polarity (PinPol)**—Bit 13. This bit is not used because LINT interrupts are delivered by HyperTransport™ messages instead of individual pins.

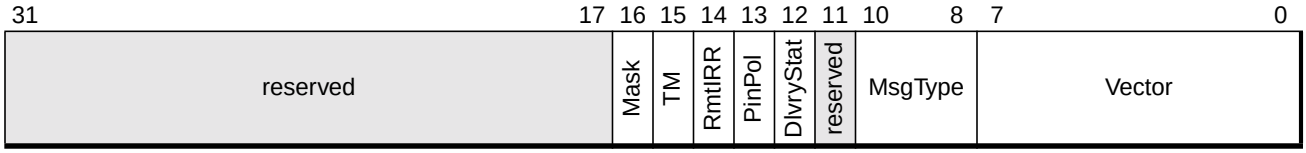
**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

**Message Type (MsgType)**—Bits 10–8. Only Message Types 000b (Fixed), 100b (NMI), and 111b (External Interrupt) are legal.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 7.8.19 Local Interrupt 1 (Legacy NMI) Local Vector Table Entry

### LINT1_LVT Register — Offset 360h



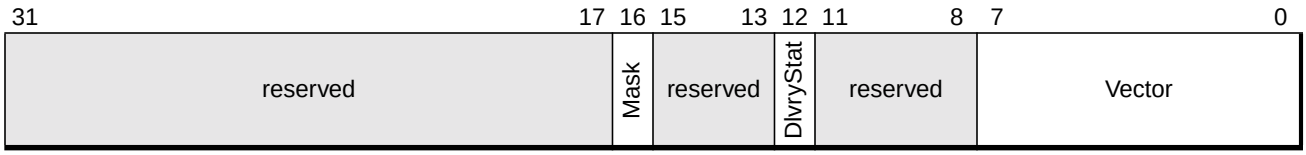

| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–17 | reserved | | R/O | 0000h |
| 16 | Mask | Mask | R/W | 1 |
| 15 | TM | Trigger Mode | R/W | 0 |
| 14 | RmtIRR | Remote IRR | R/O | 0 |
| 13 | PinPol | Pin Polarity | R/W | 0 |
| 12 | DlvryStat | Delivery Status | R/O | 0 |
| 11 | reserved | | R/O | 0 |
| 10–8 | MsgType | Message Type | R/W | 000b |
| 7–0 | Vector | Vector | R/W | 00h |

### Field Descriptions

These fields are the same as those for Local Interrupt 0.

## 7.8.20 Error Local Vector Table Entry

### ERROR_LVT Register — Offset 370h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–17 | reserved | | R/O | 0000h |
| 16 | Mask | Mask | R/W | 1 |
| 15–13 | reserved | | R/O | 000b |
| 12 | DlvryStat | Delivery Status | R/O | 0 |
| 11—8 | reserved | | R/O | 0h |
| 7–0 | Vector | Vector | R/W | 00h |

### Field Descriptions

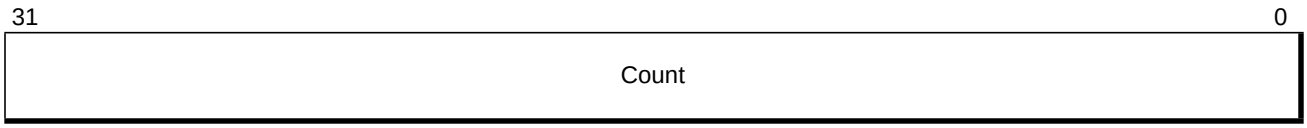
**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 7.8.21 Timer Initial Count Register

### INIT_CNT Register — Offset 380h



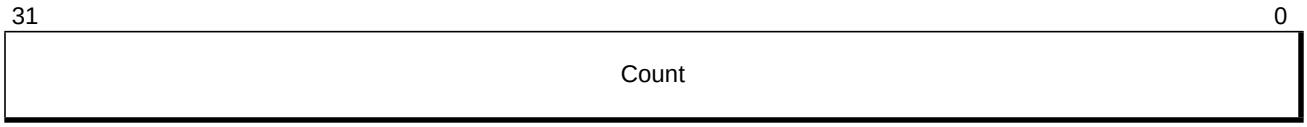

| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–0 | Count | Initial Count Value | R/W | 0000_0000h |

### Field Descriptions

**Count (Count)**—Bits 31–0. This field contains the value copied into the current count register when the timer is loaded or reloaded.

## 7.8.22 Timer Current Count Register

### CURR_CNT Register — Offset 390h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–0 | Count | Current Count Value | R/O | 0000_0000h |

### Field Descriptions

**Count (Count)**—Bits 31–0. This field contains the current value of the counter.

## 7.8.23 Timer Divide Configuration Register

### TIMER_DVD_CFG Register — Offset 3E0h




| Bit | Name | Function | R/W | Reset |
|---|---|---|---|---|
| 31–4 | reserved | | R/O | 0000_00h |
| 3 | Div[3] | Div[3] | R/W | 0 |
| 2 | reserved | | R/O | 0 |
| 1–0 | Div[1–0] | Div[1–0] | R/W | 00b |

### Field Descriptions

**Div[3] and Div[1–0]**—Bits 3 and 1–0. The Div bits are encoded as follows:

| Div[3] | Div[1–0] | Resulting Timer Divide |
|---|---|---|
| 0 | 00 | 2 |
| 0 | 01 | 4 |
| 0 | 10 | 8 |
| 0 | 11 | 16 |
| 1 | 00 | 32 |
| 1 | 01 | 64 |
| 1 | 10 | 128 |
| 1 | 11 | 1 |

# 8 HyperTransport™ Technology Configuration and Enumeration

An AMD Opteron™ processor or AMD Athlon™ 64 processor is a node connected to other nodes with coherent HyperTransport™ links. Function 0 HyperTransport technology configuration register values for each node must be initialized. A node is identified in the Node ID register (Function 0, Offset 60h) with a number from 0 to 7. Routing tables are used by the node to decide whether to accept a packet and/or forward it through any or all of its HyperTransport links.

Three examples of coherent HyperTransport technology initialization sequence are shown in this chapter: 1-node initialization, 2-node initialization, and a dynamic coherent initialization that can enumerate any number of nodes.

## 8.1 Initial Configuration Steps

For coherent HyperTransport technology initialization, the following steps are required before enumeration:

1. Coherent HyperTransport initialization is only performed by Node 0 (BSP). All the other nodes (APs) should bypass the HyperTransport initialization process.
2. Clearing Function 0, Offset 6Ch[0] enables the routing table for Node 0, and allows access to the memory controller on Node 0.
3. Node 0 is by definition connected to the HyperTransport I/O Hub, which means that Node 0 owns the compatibility chain. The link number that connects to HyperTransport I/O Hub should be written to Function 0, Offset 64h[SbLink].
4. Count coherent HyperTransport links on Node 0.

To perform the step 4 above, it is necessary to detect whether each HyperTransport link on a node is connected, and if the connected link type is coherent or noncoherent. The AMD Opteron™ processor supports up to three links, and the AMD Athlon™ 64 processor supports one link. For each HyperTransport link, the following steps are performed to detect the link connection status and type:

1. Check whether the Link Connect Pending bit is set in Function 0, Offset 98h[4].
2. If the Link Connect Pending bit is clear, check whether the Link Connected bit is set in Function 0, Offset 98h[0].
   If the Link Connected bit is set, the testing port is connected to other node.
3. Check whether the Initialization Complete bit is set in Function 0, Offset 98h[1].
4. If the bit is set, check the Non Coherent bit in Function 0, Offset 98h[2].
   The bit value 0 indicates a coherent HyperTransport link, while value 1 indicates a noncoherent link.

5. Record the number of coherent and noncoherent ports and their respective port numbers.

## 8.2 One-Node Coherent HyperTransport™ Technology Initialization

The number of nodes that exist in the system is determined on page 203. The following configuration steps should be executed in single (one-node) processor systems:

1. Set Function 0, Offset 68h[LimitCldtCfg] bit to enable limiting the extent of coherent HyperTransport configuration space based on the number of nodes in the system.
2. Disable read/write/fill probes in Function 0, Offset 68h[10, 3:0], since single processor systems do not need probes.

## 8.3 Two-Node Coherent HyperTransport™ Technology Initialization

For two-node systems, Node ID and Routing Tables need to be written for both nodes, as follows.

1. Initialize Node 0 Routing Table (RT) rows 0 and 1. Since the coherent HyperTransport link number on Node 0 is already known, RT can be generated using the backbone-first algorithm. See "Algorithm" on page 207 for the backbone-first routing algorithm.
2. Initialize Node 0 RT row 7. Since Node Id of Node 1 is 7 by default, this ID will be used temporarily to configure Node 1. RT row 7 is the same as RT row 1.
3. Verify if Node 0-to-Node 7 link is established by reading Vendor ID/Device ID of Node 7.
4. Detect the connected coherent HyperTransport link number on Node 7 by reading Default Link in Function 0, Offset 6Ch[3:2].
5. Initialize Node 7 RT row 0 and 1. Since the link number on Node 7 is known, RT can be generated using the backbone-first algorithm.
6. Renumber Node 7 to Node 1 by programming Function 0, Offset 60h[2:0].
7. Reset row 7 on Node 0 to the default value.
8. Write CPU Count and Node Count in Function 0, Offset 60h[19:16, 6:4].
9. Set Function 0, Offset 68h[LimitCldtCfg] bit for Node 0 and Node 1.

# 8.4 Generic HyperTransport™ Technology Configuration

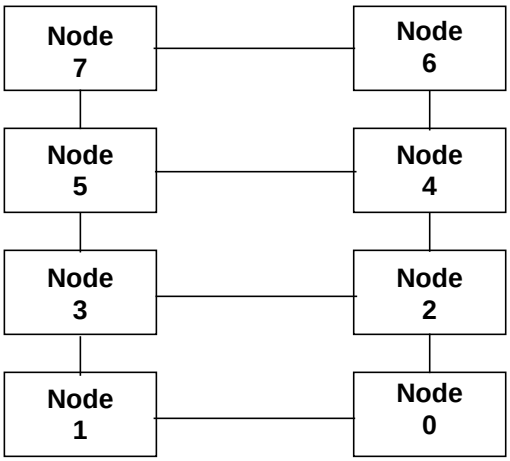
This section presents a complete dynamic algorithm for initialization of the coherent HyperTransport technology fabric. This algorithm describes configuration register initialization for proper routing of HyperTransport technology traffic through the coherent fabric.

First, system configuration assumptions are made. Second, the fabric initialization algorithm is presented. Last, a sub-algorithm for writing rows of routing tables is described. See Chapter 3, "Memory System Configuration" for details about configuration registers.

The described algorithm assumes one of the following system configurations:

| | | | | |
|---|---|---|---|---|
| 1-node -> 1x1 | 2-node -> 2x1 | 4-node -> 2x2 | 6-node -> 2x3 | 8-node -> 2x4 |

For example, the 8-node system diagram is shown in Figure 3, where the node numbers are assigned by the algorithm. Other configurations may be preferable to the one shown in Figure 3, but they are not addressed.




***Note:*** *This is an 8-node system configuration. On the right side, from bottom to top, the nodes are numbered 0, 2, 4, and 6, while on the left, also from bottom to top, they are 1, 3, 5, and 7. The horizontal links (between nodes 0 and 1, 2 and 3, 4 and 5, and 6 and 7) are called "across". The link from 0 to 2 is "up" (similarly for other links from i to i+2) and from 2 to 0 is "down" (similarly for other links from i+2 to i). The slot formation for 2, 4, or 6 nodes is obtained by deleting 2-node horizontal rows, starting at the top of the figure.*

### Figure 3. An Eight-Node Configuration

The algorithm determines node numbers and initializes *backbone-first* routing tables, which provide performance that is reasonably close to optimal in numerous settings. The idea of the backbone-first

algorithm is explained below and refers only to the request and broadcast tables. The response tables produce paths in the reverse order from paths produced by the other two tables. If the nodes and links are laid out in one of the *slot* or *planar* configurations, then this algorithm will produce the expected node numbers.

In the following description, a read or write to a register on Node *n* is always accomplished by an access to the register offset in configuration space, function 0, device 24+*n*, where *n* is the target Node ID. For example, to write a value into the Node ID register of Node *n*, function 0, device 24+*n*, offset 60h is used to form the address. Until a node's routing tables have been written, its routing table causes all messages to all devices from 24 to 31 to be accepted locally.

Configuration is viewed as a ladder extending upward, as shown in the Figure 3, where the backbone (upward) direction is perpendicular to the "rungs". A data structure in table format for each node has information about adjacent links: "up", "across", and "down". The optimal routing table is backbone-first, since request paths follow along the length of the ladder, only going along a rung at the very end of the path (if necessary). This table is kept in the BSP address space, although it contains information for all nodes.

Section "Writing Routing Table Rows" on page 213 describes how to use link information—"up", "across", "down"—to create the backbone-first routing tables used in various steps of the algorithm.

The BSP will execute code from the firmware ROM to initialize the coherent HyperTransport technology fabric.The assumption is that each node has the following registers that can be accessed by the BSP.

1. NodeId[2:0], which has default value of 0 on the BSP node and 7 on the other nodes.
2. Routing tables that indicate to which links to route a packet and whether to route the packet to the internal node based on the destination NodeID of the packet. These have default value "route to this node", which results in accepting all commands and not forwarding them.
3. A RoutingTableDisable (RTD) bit forces all commands to be accepted and all responses to be returned back to the link that delivered the request, regardless of the routing tables. If a processor on that node is sending messages through the crossbar, then this assumption could be incorrect (see step 6 below).
4. DefaultLink field (Function 0, Offset 6Ch) that records the link from which the last request was received.
5. A LinkType register indicating how many connected links the node has and whether the links are coherent or noncoherent.
6. A RequestDisable bit that prevents the node from making a request. This bit is cleared on the BSP coming out of microcode reset, and should be set on all other nodes.

The code below is only executed on the BSP. The following assumptions are made for each node.

1. The NodeID is ignored when determining where to route a command (including configuration reads and writes). Determining routing and acceptance is only based on the node's routing table, the destination node in the command, and the destination unit in the command.

2. The result of a read or write to configuration space is independent of the destination node in the command. For example, if a node accepts a command that reads its Node ID, then it will respond with the value of its Node ID register even if the destination was a device not corresponding to that Node ID.

Node LDT*i* Type registers (*i*=0,1,2) are read to determine its connected coherent HyperTransport links not marked as UP.

The following steps are grouped into three blocks, starting with Nodes 0, 1, 2 and 3, then Nodes 4 and 5, and finally Nodes 6 and 7.

## 8.4.1 Algorithm

Pseudo-code for fabric initialization is described in this section. Steps taken after the initialization of Node IDs and routing tables are not described.

Subroutine fill_routing_table(Node*N*, row*r*), described in "Writing Routing Table Rows" on page 213, writes row *r* of the routing table for Node *N*. On a few occasions, a row of a routing table is written directly without using this subroutine.

Northbridge registers on nodes other than Node0 should not be accessed before microcode on AP processors has set the RequestDisable bit. Thus, the first access to an AP node is a read of the RequestDisable bit (Function 0, Offset 6Ch). This bit should be polled until it is set. A possible first access to each node is marked with "**(first_access)**". The value returned for the Function 0, Offset 6Ch register read can be 0FFFFFFFFh, indicating a time-out. In that case, the link is treated as not present.

The algorithm comments are in parentheses.

### BLOCK A: Complete the first four nodes, stopping short if there are fewer than four nodes

A1. If Node0 has no coherent HyperTransport links other than possibly one marked as UP, then it is a single-node system. There is no need to change the routing tables or set the NodeId, except for the case where a link *L* is marked as UP. In that case, rows 0 and 1 of Node0's routing table should be written after marking the UP link as "across" and marking one of the other two links as "up" and the other as "down." Exit.

EXIT IF ONLY ONE NODE OR UNIPROCESSOR.

A2. If Node0 has only one coherent HyperTransport link, which is not UP, then it is a two-node system. The following steps should be executed.

A2.1. Mark the coherent link of Node0 as "across" and mark the other two links of Node0 as "down" and "up", arbitrarily.

A2.2. Call fill_routing_table(Node0,row*i*) for *i*=0,1.

A2.3. Write a NodeId of 1 to Device 25 **(first_access)**.

A2.4. Read the default link of Node1, and mark it as "across". Mark the remaining two links of Node1 as "down" and "up", arbitrarily.

A2.5. Call fill_routing_table(Node1,row*i*) for *i*=0,1.

A2.6. Clear the RTD bit for Node1.

A2.7. Exit.

EXIT IF ONLY 2 NODES.

(It is known at this point that Node0 has exactly two coherent links, since one link is an noncoherent link leading to the HyperTransport I/O Hub.)

A3. For each coherent link *L* of Node0, determine the number of coherent links in the node on the other side of L, as follows.

A3.1. Write row 7 of Node0's routing table to a value that causes requests and responses destined for Device 31 (Node7) to be routed through link *L*.

A3.2. Read the number of coherent links on Device 31 **(first_access),** and save the association of link *L* with this number.

(It is known at this point that Node0 has exactly two coherent links, since one link is a noncoherent link to the HyperTransport I/O Hub. It is assumed that there are minimum four nodes, where each of the two links connects Node0 to a node with minimum two coherent links. The node across from Node0 has exactly two coherent links, since it is in the "corner" of the configuration. It is also known whether the system is limited to four nodes: there are more than four nodes if and only if one of the nodes connected to Node0 has three coherent links (the one labeled as Node2 at the end of the algorithm).)

A4. Mark the coherent links of Node0 and write rows 0 to 2 of its routing table.

A4.1. Mark as "across" the coherent link from Node0 that points to a node with only two coherent links (as saved in Step A3). If there are two such nodes, i.e., if it is a four-node system, then the "across" link is set to be the smaller of the two coherent link numbers of Node0. (Note that the numbering will be correct for both the *slot* and *planar* four-node configurations.)

A4.2. Mark as "up" the other coherent link of Node0, and as "down" the noncoherent link of Node0.

A4.3. Call fill_routing_table(Node0,row*i*) for *i*=0,1,2. (A temporary value for row 3 will be written in Step A7.1, and row 2 is the last one written here.)

A5. Complete the node across from Node0 by writing its NodeID to 1, marking its links, writing its routing table, and clearing its Routing Table Disable (RTD) bit.

A5.1. Write a NodeID value of 1 to Device 25.

A5.2. Read the default link at Node1 (which will be the one pointing to Node0). Then mark that link as "across".

A5.3. Mark the remaining connected coherent link of Node1 as "up" and its last unmarked link as "down".

A5.4. Call fill_routing_table(Node1,row*i*) for *i*=0,1,2,3.

A5.5. Clear Node1's RTD.

A6. Write a NodeID of 2 to the node "up" from Node0, and then mark "down" link of Node2.

A6.1. Write a NodeID value of 2 to Device 26.

A6.2. Mark the link at Node2 down to Node0 as "down", by reading the default link as in Step A5.2 above.

A7. Write a NodeID of 3 to the node "up" from Node1, and mark its link to Node1 as "down", as follows.

A7.1. Write row 3 of Node0's routing table so that messages destined for Node3 are sent "across".

A7.2. Mark the link at Node3 down to Node1 as "down" by reading the default link of Device 27 **(first_access)**.

A7.3. Write a NodeID of 3 to Device 27.

A8. Call fill_routing_table(Node0,row3).

(At this point Node0 initialization is complete: it has the correct NodeID and routing table, and its coherent links are appropriately marked "across" and "up". Node1 initialization has also been completed (Step A5). Node2 and Node3 have Node IDs and still have their RTD bits set.)

A9. If it is a four-node system (because Node2 has two coherent links, as discussed in the comment of Step A3), then the following steps should be executed.

A9.1. Mark as "across" the remaining coherent link of Node2 (the one not marked as "down"), and then as "up" the final link of Node2.

A9.2. Call fill_routing_table(Node2,row*i*) for *i*=0 to 3 and then clear Node2's RTD.

A9.3. Read the default link of Node3 and mark it as "across". (Since Node0 has a "backbone-first" routing table, the path of that read from Node0 to Node3 is by Node2.)[1]

A9.4. Set the remaining link of Node3 (the one not set just above or in Step A7.2) to "up".

A9.5. Call fill_routing_table(Node3,row*i*) for *i*=0 to 3 and then clear Node3's RTD.

A9.6. Exit.

EXIT IF ONLY 4 NODES.

(If this point is reached, then there are more than four nodes, of which the first four rows of Node0 and Node1 routing tables (Steps A4.3/A8 and A5, respectively) are complete. Node2 is partly complete (Step A6): it has its NodeID of 2 and its link to Node0 has been marked "down". The NodeID field has been written for Node3 and its "down" link has been marked (Step A7). Also, RTD is still set for Node2 and Node3. The other two Node2 links need to be marked and routing table initialized, and Node3 handling needs to be finished, before moving to other nodes (starting with Step B1 below).)

A10. Call fill_routing_table(Node1,row*i*) for *i*=4,5 and call fill_routing_table(Node0,row4).

A11. Finish marking the links of Node2, write its routing table, and clear its RTD.

A11.1. Mark the lower number of the remaining two links of Node2 as "across" and the other as "up". This is a guess.

A11.2. Call fill_routing_table(Node2,row*i*) for *i*=0,3 and clear Node2's RTD.

A11.3. Check the guess and modify it if necessary.

A11.3.1. Read the NodeID of Device 27. (Node2 will route this through the link marked as "across", which was a guess.)

A11.3.2. If the NodeID just read is 3, then the guess was correct; call fill_routing_table(Node2,row*i*) for *i*=1,2,4,5. Otherwise, switch the marking of the "across" and "up" links of Node2, and then call fill_routing_table(Node2,row*i*) for *i*=1 to 5. (Note that row 0 does not need to be modified either way. Also, if the read times out, then the result will be 7. In that case, it can be assumed that the read was done through the link pointing up.)

A12. Finish marking the links of Node3, write its routing table, and clear its RTD.

A12.1. Read the default link register of Node3. (Now that Node0's and Node2's routing tables

1. Steps A9.3 and A9.4 could possibly be replaced by a single step that marks the remaining coherent link of Node3 as "across". The following case: Node2 and Node3 both have 3 coherent links, and one of Node2's links is not connected, results in the four-node case. Similar cases are ignored in other places.

are correctly defined, that link connects across to Node2.) Mark that link as "across", and mark the remaining unmarked link of Node3 as "up".

A12.2. Call fill_routing_table(Node3,row*i*) for *i*=0 to 5 and clear Node3's RTD.

(It is known at this point that there are more than four nodes. Nodes 0 through 3 have now received their Node IDs and rows 0 through 5 of their routing tables, and their routing tables are enabled (i.e., their RTD bits are cleared). Row 5 of Node0's routing table has not been written yet, since it can change in Step B3.1.)

### Block B: Complete Node4 and Node5

B1. Write the NodeID of Node4 and mark its "down" link.

B1.1. Write a NodeID of 4 to Device 28 **(first_access)**.

B1.2. Read the default link of Device 28 (Node4) and mark it as "down".

B2. If Node4 has only two coherent links, then complete both Node4 and Node5 and then exit.

B2.1. Mark the unmarked coherent link of Node4 as "across", and then mark its remaining link as "up".

B2.2. Call fill_routing_table(Node4,row*i*) for *i*=0 to 5 and clear Node4's RTD.

B2.3. Call fill_routing_table(Node0,row5).

B2.4. Write a NodeID of 5 to Device 29 **(first_access)**.

B2.5. Read the default link of Node5 (Device 29), and mark it as "across".

B2.6. Mark the remaining coherent link of Node5 as "down" [2], and then mark the final link of Node5 as "up".

B2.7. Call fill_routing_table(Node5,row*i*) for *i*=0 to 5 and clear Node5's RTD.

B2.8. Exit.

EXIT IF ONLY 6 NODES.

(At this point, Nodes 0 through 3 are initialized, Node4 has its NodeID, the "down" link of Node4 is marked, and Node4 is known to have three coherent links. Also, the RTD bit is set for Node4 and all greater nodes. Row 5 of Node0's routing table is not written.)

2. If Node5 has three active coherent links, which can happen if Node4 has three links marked as active coherent links but the one "up" is not functioning, then a temporary routing table row change as in Step B3 can be made.

B3. Write a NodeID of 5 to the node across from Node4 and mark that node's "down" link. (The "across" link of Node4 has not been marked yet, and messages are not routed through Node4 to Node5.)

B3.1. Write row 5 of Node0's routing table to point "across" for requests (so that requests from Node0 to Device 29 will follow the path 0->1->3->*N* where *N* is the node "up" from Node3.)[3]

B3.2. Read the default link of Device 29 **(first_access)** and mark that link as "down" for Node5.

B3.3. Write a NodeID of 5 to Device 29.

B4. Call fill_routing_table(Node0,row*i*) for *i*=5,6,7 and call fill_routing_table(Node*N*,row*i*) for *N*=1,2,3 and *i*=6,7.

B5. Finish marking the links of Node4, write its routing table, and clear its RTD. This step and the next are completely analogous to Steps A11 and A12.

B5.1. Mark the lower number of the two unmarked links of Node4 as "across" and the other as "up". This is just a guess.

B5.2. Call fill_routing_table(Node4,rowi) for *i*=0,5 and clear Node4's RTD.

B5.3. Check if the guess in Step B5.1 was correct, and if not, then correct it.

B5.3.1. Read the NodeID of Device 29. (Node4 will route this write through the link marked as "across", which was a guess.)

B5.3.2. If the NodeID read is 5, then the guess was correct; call fill_routing_table(Node4,row*i*) for *i*=1 to 4 and for *i*=6,7. Otherwise, switch the marking of the "across" and "up" links of Node4, and then call fill_routing_table(Node4,row*i*) for *i*=1 to 7. (Row 0 need not be re-written. And the case of timing should be OK, as in Step 11.3.2.)

B6. Finish marking the links of Node5, write its routing table, and clear its RTD.

B6.1. Read the default link register of Node5. Mark that link as "across", and mark the remaining unmarked link of Node5 as "up".

B6.2. Call fill_routing_table(Node5,row*i*) for *i*=0 to 7 and clear Node5's RTD.

3. Step B3 approach will cause responses to take the path 5->3->2->0, which is not the reverse of the request path from 0->1->3->5. That is acceptable, although the tracing will not function equally well. This can be fixed by modifying Node2's routing table instead of Node 0's, so that the request path is 0->2->3->5.

(It is known at this point that Node4 has 3 coherent links, so there are presumably 8 nodes.[4] Rows 0 to 7 have been written for the routing tables of Nodes 0 to 5, and the RTD bits are clear for those nodes.)

### Block C: Complete Node6 and Node7

C1. Write to Device 30 a NodeID of 6 **(first_access)**.

C2. Read the default link of Device 30 (Node6) and mark it as "down".

C3. There should be exactly one unmarked, active coherent link in Node6. Mark it as "across", and then mark the remaining link as "up".[5]

C4. Call fill_routing_table(Node6,row*i*) for *i*=0 to 7 and clear Node6's RTD.

C5. Read the default link of Device 31 **(first_access)** and mark it as "across". (NodeID does not have to be written, since the reset microcode has already written a NodeID of 7.)

C6. There should only be one unmarked active coherent link on Node7. Mark this link as "down", and then mark the remaining link as "up".

C7. Call fill_routing_table(Node7,row*i*) for *i*=0 to 7 and clear Node7's RTD.

### Completion

At this point, all nodes have their Node IDs and routing tables. Also, it is known which NodeID was last written; this is the maximum NodeID, M. Adjust the broadcast tables of nodes M and M-1 as it is described in "Writing Routing Table Rows" on page 213. Then write every node's NodeID register, in particular its NodeCount.

During the execution of this algorithm, configuration space accesses are sent to corresponding coherent nodes, not to the compatibility bus. Bit LimitCldtCfg (Function 0, Offset 68h) is set near the end to enable range based on node count. For example, if there are two nodes (Devices 24 and 25), then accesses to Devices 26 through 31 will go to the compatibility bus.

Finally, the RequestDisable bit is cleared at each AP processor.

Before deciding the maximum NodeID and adjusting the broadcast tables, functionality of all links must be verified.

#### 8.4.1.1 Writing Routing Table Rows

Routine fill_routing_table(Node*N*,row*r*) which writes a given row *r* of Node *N*'s routing table is described in this section. The following assumption are made: the routing tables for all nodes on the

4. A check that there are 8 nodes can be done by checking that the node "up" from Node4 has two connected coherent links. If the check fails, jump from Step C5 to 6-node case.
5. If the case that there is no unmarked active coherent link in Node6 is considered, the algorithm can be exited with a 6-node system.

path up to Node *N* (not including Node *N*) have been written, and the Node *N* links have been correctly marked as "up", "down", or "across".

Statement "put a 1 in the P position", means that if link *L* is associated with P at node *N* (where P is "up", "down", or "across"), then bit position *L*+1 is assigned the value 1.

REQUEST TABLE. Row r of the "backbone-first" request table is computed as follows.

If *r* = *N*, then the value is 0001b. Else if (*r* XOR 1) = N, then put a 1 in the "across" position. (This is testing that *r* and *N* differ in the unit digits only, so they are across from each other.) Else if (*r* < *N*), then put a 1 in the "down" position. Else put a 1 in the "up" position.

RESPONSE TABLE. For the response table, the following rule is used ("short-first" or "rung-first" rather than "backbone-first").

If *r* = *N*, then the value is 0001b. Else if ((*r* XOR *N*) AND 0001b) != 0, i.e. *r* and *N* are in different "rungs", then put a 1 in the "across" position. Else if (*r* < *N*), then put a 1 in the "down" position. Else put a 1 in the "up" position.

BROADCAST TABLE. The final value for each row of Node *N*'s broadcast routing table must comply with the following requirements:

1. If *N* is 0 or 1 then the "down" bit is 0.
2. Let M be the maximum node number. If *N* is M or M-1 then the "up" bit is 0. Note: Since M is not always known at the time routing tables are written, this step should be performed at the end of the process.
3. The least significant bit "route to this node" is 1.

The rules for constructing row *r* of the broadcast table for Node *N* that should be applied before finalizing the result of 1., 2., and 3. above. For the broadcast table, unlike the request and response tables, *r* represents the source of the message, not the destination.

If *r* = *N*, then the value is 1111b. Else if ((*r* XOR *N*) AND 0001b) != 0, i.e., *r* and *N* are in different "rungs", then return 0001b. Else if *r* < *N*, put a 1 in the "up" and "across" positions. Else put a 1 in the "down" and "across" positions.

# 9 Power and Thermal Management

AMD Athlon™ 64 and AMD Opteron™ processors support ACPI-compliant power management for all classes of systems from notebook PCs to multiprocessor servers. Table 40 lists various levels of power management. Table 41 on page 216 lists ACPI-state support by system class. This chapter covers processor-level power management and processor-specific aspects of system-level power management.

**Table 40. Power Management Categories**

| Power Management Category | Comments |
|---|---|
| System Level | Coarse level power management applied to the entire system, typically after a predefined period of inactivity, or in response to a user action such as pressing the system power button.<br>• System Sleep States (ACPI-defined S-states) |
| Processor Level | • Processor Power States (ACPI-defined C-states)<br>• Processor Performance States (ACPI-defined P-states)<br>• Software Transparent Power Management (Hardware enforced throttling). |
| Device Level | • Device Power States (ACPI-defined D-states)<br>• Device Performance States<br>• Software Transparent Power Management. |
| Bus Level | • Bus Power States<br>• Software Transparent Power Management |
| Sub-system Level | • Typically not used in PC systems, and beyond the scope of this document. |

**Table 41. ACPI-State Support by System Class**

| ACPI State Support | Mobile Systems | Uniprocessor Desktop PCs | Multiprocessor Systems |
|---|---|---|---|
| G0/S0/C0: Working | Yes | Yes | Yes |
| G0/S0/C0: Processor performance state (P-state) transitions under OS control. | Yes | Yes[1] | Not with current operating systems |
| G0/S0/C0: Thermal clock throttling (I/O hub hardware enforced) | Yes | Yes | Revision B:No[1]<br>Revision C:Yes |
| G0/S0/C0: Thermal clock throttling (operating system enforced) | Yes | Yes | No |
| G0/S0/C1: Halt | Yes | Yes | Yes |
| G0/S0/C2: Stop Grant Caches snoopable | Yes[1,2] | No | No |
| G0/S0/C3: Stop Grant Caches not snoopable | Yes[1,2] | No | No |
| G1/S1: Stand By (Powered On Suspend) | Yes | Yes | Yes |
| G1/S3: Stand By (Suspend to RAM) | Yes | Yes | Revision B:No[1]<br>Revision C:Yes |
| G1/S4: Hibernate (Suspend to Disk) | Yes | Yes | Yes |
| G2/S5: Shut Down, Turn Off (Soft Off) | Yes | Yes | Yes |
| G3 Mechanical Off | Yes | Yes | Yes |

***Notes:***

1. *Not supported by some revisions of the silicon. Refer to Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors, order# 25759.*
2. *See "Advanced Programmable Interrupt Controller (APIC)" on page 181.*

# 9.1 Stop Grant

The processor and chipset use HyperTransport™ technology STPCLK and Stop Grant system management messages to sequence into all power management states except for Halt. These messages carry a 3-bit System Management Action Field (SMAF) to differentiate the various reasons for placing the processor into the Stop Grant state. Table 42 maps Stop Grant SMAF codes to ACPI states and actions that the BIOS is required to configure the processor to take for each ACPI state.

**Table 42. Required SMAF Code to Stop Grant Mapping**

| Reason for STPCLK Message | SMAF Field of STPCLK and Stop Grant Messages | Mechanism that Forces the I/O Hub to Send the STPCLK Message | Processors Programmed Response after Entering the Stop Grant State (See Table 52 Power Management Control Registers.) |
|---|---|---|---|
| C2 Stop Grant Caches Snoopable | 000b | Sent in response to a read of the ACPI-defined P_LVL2 register. | Ramp the CPU clock grid down when no probes need to be serviced. (CPU Low Power Enable) |
| C3 Stop Grant Caches not Snoopable. Used only by mobile systems. | 001b | Sent either in response to a read of the ACPI-defined P_LVL3 register or in response to a write to the Link Frequency Change and Resize LDTSTOP_L Command register in the I/O hub. | Ramp the CPU clock grid frequency down. (CPU Low Power Enable)<br>After LDTSTOP_L is asserted, place memory into self-refresh and ramp down the processor host bridge and memory controller clock grid. (Northbridge Low Power Enable) |
| VID/FID Change<br>Or<br>Link Width/ Frequency changes | 010b | Sent either in response to a VID/FID Change special cycle from the processor or in response to a write to the Link Frequency Change and Resize LDTSTOP_L Command register. | After LDTSTOP_L assertion, place memory into self-refresh, ramp the processor clock grids down, then drive new FID values to PLL. (CPU Low Power Enable, Northbridge Low Power Enable, FID/VID Change Enable) |
| S1 Sleep state | 011b | Sent in response to writing the S1 value to the SLP_TYP[2:0] field and setting the SLP_EN bit of the ACPI-defined PM1 control register in the I/O hub. | Same response as C3. |
| S3 Sleep state | 100b | Sent in response to writing the S3 value to the SLP_TYP[2:0] field of the PM1 control register. | Same response as C3. Additionally, after LDTSTOP_L has been asserted, the I/O hub will power off the main power planes. |
| Throttling | 101b | Occurs based upon hardware initiated thermal throttling or ACPI-controlled throttling. | Ramp down the CPU grid by the programmed amount. (CPU Low Power Enable) |
| S4/S5 | 110b | Sent in response to writing the S4 or S5 value to the SLP_TYP[2:0] field of the PM1 control register. | Same response as S3 for processor. Additionally, all power will be removed from processor during S4 and S5. |

**Table 42. Required SMAF Code to Stop Grant Mapping (Continued)**

| Reason for STPCLK Message | SMAF Field of STPCLK and Stop Grant Messages | Mechanism that Forces the I/O Hub to Send the STPCLK Message | Processors Programmed Response after Entering the Stop Grant State (See Table 52 Power Management Control Registers.) |
|---|---|---|---|
| Reserved for use by processor | 111b | No I/O hub STPCLK message uses this SMAF code. The power management register field corresponding to this SMAF code is used by the processor in response to executing the Halt instruction. | Same response as C2, except a Halt special cycle is broadcast. |

# 9.2 C-States

As Table 40 on page 215 indicates, the processor supports ACPI-defined processor power states, which are referred to as C-states. Table 41 on page 216 indicates which C-states are supported by system class. The operating system will place the processor into C-states when the processor is idle an operating-system-determined percentage of the time.

## 9.2.1 C1 Halt State

C1 is the Halt state. C1 is entered after the HLT instruction has been executed. The BIOS configures the processor to enter a low-power state during C1, in which the CPU core clock grid is ramped down from its operating frequency. See Table 52 on page 242.

## 9.2.2 C2 and C3

C2 is the Stop Grant state in which the processor caches can be snooped. When the processor is in the Stop Grant state and there is no probe activity to the processor's caches, the CPU core clock grid is ramped down from its operating frequency. For more information, see Table 52 on page 242.

C3 is the Stop Grant state in which the processor's caches cannot be snooped. The chipset asserts LDTSTOP_L during the C3 state. The chipset may require BIOS to enable LDTSTOP_L assertion for the C3 state. After LDTSTOP_L assertion, the processor's system memory is placed into self-refresh mode and the processor's clock grids, including the host bridge/memory controller clock grid, are ramped down from their operational frequency. For more information, see Table 52 on page 242.

# 9.3 Throttling

The BIOS must declare a DUTY_WIDTH value of zero in the Fixed ACPI Description table if the system does not support the _PSV object as part of a processor ACPI thermal zone. Throttling is not

supported by some revisions of the silicon. Refer to *AMD Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25759.

Chipsets must not be configured to assert LDTSTOP_L during throttling.

## 9.4 Processor ACPI Thermal Zone

This section will be part of a future revision of this document.

## 9.5 Processor Performance States

Processor performance states (P-states) are valid operating combinations of processor core voltage and frequency. The hardware supporting P-state transitions in AMD processors is referred to as AMD PowerNow!™ technology for mobile systems and AMD Cool'n'Quiet™ technology for desktop systems. In this document all descriptions of AMD PowerNow!™ technology, software and driver apply to AMD Cool'n'Quiet™ technology, software, and driver.

- For operating systems without native support for P-state transitions (legacy operating systems) using Revision C and subsequent processors, AMD PowerNow! software is used to perform P-state transitions. In this case, a BIOS generated Performance State Block (PSB) is used by AMD PowerNow! software to determine what P-states are supported by the processor in the system.
- For operating systems that have native support for P-state transitions using Revision C and subsequent processors, a processor-specific driver is used by the operating system to make P-state transitions. In this case, BIOS-provided ACPI-defined P-state objects inform the operating system that P-state transitions are supported by the system, which P-states are supported, and when given P-states are available for use by the operating system.

The processor data sheet specifies the valid P-states for each processor. The BIOS vendor must only use P-states defined in that document for a given processor when building the ACPI P-state objects and PSB for use by higher-level software.

Processors that support P-state transitions provide MSR C001_0041h (FIDVID_CTL) and MSR C001_0042h (FIDVID_STATUS) for initiating P-state transitions and verifying that they are complete. If these two MSRs are accessed in a processor that does not support P-state transitions, then a general protection fault occurs.

Chipsets provide support for P-state transitions. Refer to the chipset documentation for information on chipset configuration and P-state-related considerations.

### 9.5.1 BIOS Requirements for P-State Transitions

P-state transitions can be used only if they are supported by the processor and by the system.

BIOS is required to:

- Determine that the processor supports P-state transitions.
- Have a valid set of P-states for the processor, based on the processor data sheet.
- Take chipset or system limitations into account before providing the ACPI-defined P-state objects described in 9.6 on page 230, or the PSB described in 9.7 on page 239 of this document.
- If P-states are supported and a Revision C or higher revision processor is present, BIOS is required to provide both:
  - ACPI-defined P-state objects for operating systems that support native P-state control.
  - a PSB in support of AMD PowerNow! software with legacy operating systems.

The BIOS must not provide the ACPI P-state objects or the PSB if:

- The processor does not support P-state transitions.
- The processor is Revision B.
- The chipset or system does not support P-state transitions.
- The BIOS does not have a set of valid P-states derived from the processor data sheet.

#### 9.5.1.1 Step 1: Determine Processor Support for P-State Transitions

The BIOS determines if a processor supports P-state transitions by executing the CPUID extended function 8000_0007h.

If the value returned in EDX[2:1] is:

- 11b, then the part is capable of performing P-state transitions.
- 00b, then the part is *not* capable of performing P-state transitions.

Refer to the following documents regarding CPUID and determining which processor is in the system:

- *AMD Processor Recognition Application Note*, order# 20734
- *CPUID Guide for the AMD Athlon™ 64 and AMD Opteron™ processors,* order# 25481

#### 9.5.1.2 Step 2: Match the Processor to a Valid Set of P-States

If CPUID extended function 8000_0007h returned EDX[2:1] = 11b, the BIOS must correlate the processor in the system to a valid set of P-states from the processor data sheet. Each processor is uniquely identified by the following:

- CPUID extended function 8000_0001h:
  - Family number and extended family number (when family number field = 1111b).
  - Model number and extended model number (when family number field = 1111b).
  - Stepping (Revision).

- FIDVID_STATUS MSR (C001_0042h):
  - MaxVID: The nominal voltage for the maximum P-state is the voltage selected by MaxVID minus the ramp voltage offset (RVO).
  - StartVID: This corresponds to the nominal voltage at which the processor starts operation after a cold reset.
  - MaxFID: This corresponds to the frequency that the processor operates at when in the maximum performance state.
  - StartFID: This corresponds to the frequency that the processor starts operation at after a cold reset.

These seven parameters are used by the BIOS to match the processor in the system to a valid set of P-states (voltage and frequency combinations) listed in the processor data sheet.

#### 9.5.1.3 Step 3: Determine System Support for P-State Transitions

The BIOS software must also determine whether the system supports P-state transitions. The BIOS must take into account chipset/motherboard components and potential system side effects associated with P-state transitions that could preclude the use of P-state transitions.

### 9.5.2 BIOS-Initiated P-State Transitions

Processors that do not support P-state transitions and processors intended for desktop systems power up to the maximum P-state. Processors intended for mobile systems power up in the minimum P-state. BIOS can transition mobile processors from the minimum P-state to a higher P-state during the POST routine. If BIOS performs a P-state transition it must follow the P-state transition algorithm defined in 9.5.5 on page 222. The BIOS never initiates P-state transitions after passing control to the operating system.

### 9.5.3 BIOS Support for Operating System/CPU Driver-Initiated P-State Transitions

Operating systems that have native control for processor P-states exclusively use the presence of BIOS provided ACPI 2.0 defined processor P-state objects to determine if processor P-states are supported in a system. If the ACPI objects do not exist, then the processor driver is not loaded.

In operating systems that do not have native support for processor P-state transitions, an AMD defined PSB provided by the BIOS informs AMD PowerNow! technology software that P-state transitions are supported.

Once it has been determined that a processor and system support P-state transitions, the BIOS must provide both:

- ACPI-2.0-defined processor P-state objects for operating systems with native P-state support. See Section 9.6 on page 230.

- AMD-defined PSB for use with AMD PowerNow! software and operating systems that do not have native P-state support. See Section 9.7 on page 239.

### 9.5.4 Processor Driver Requirements

The processor driver that supports native operating system policy and control of P-state objects is required to use the ACPI 2.0 P-state objects as defined in this document. The processor driver is not permitted to use the legacy PSB tables described in section 9.7 on page 239.

### 9.5.5 P-State Transition Sequence

This section describes the P-state transition algorithm for the AMD Athlon™ 64 and AMD Opteron™ processors.

#### 9.5.5.1 FID to VCO Frequency Relationship

Table 43 and Table 44 provide the core frequency-to-VCO frequency relationship for AMD Athlon™ 64 and AMD Opteron™ processors. These processors do not support frequency transitions in VCO frequency steps greater than 200 MHz. The processor driver, and the AMD PowerNow! driver use these tables when making P-state transitions.

Only one P-state is allowed in Table 43. The one P-state consists of a frequency from the "Minimum Core Frequency" column in Table 43 and is the minimum P-State supported by the processor. For processors that support P-state transitions, the processor data sheet has a table of valid P-states based on ordering part number that dictates which frequency in Table 43 is used for the minimum P-state supported by the processor.

Table 43 additionally defines Portal Core Frequencies. The "Portal Core Frequencies in Table 44" column of Table 43 defines:

- The core frequencies in Table 44 to which direct transitions can be made from the minimum core frequency in Table 43.
- The core frequencies in Table 44 that support transitions to the minimum core frequency in Table 43.

The processor supports direct transitions between the minimum core frequency in Table 43 and any of the core frequencies from Table 44 listed in the Portal Core Frequencies column associated with the minimum core frequency.

If the minimum P-state from the low table (Table 43) has a VCO frequency > 1600 MHz, then the VCO/core frequency of all P- states in the high table (Table 44) must be >= the VCO frequency of the minimum P-state minus 200 MHz.

### Example 1:

Given a processor with a minimum core frequency of 800 MHz and a maximum core frequency of 2000 MHz, to transition from 800 MHz to 2000 MHz, the processor driver will (in the order specified):

1. Transition the processor from the 800 MHz core frequency to 1800 MHz core frequency. 1800 MHz is a portal frequency to and from 800 MHz as defined in Table 43, because the VCO frequency change from an 800 MHz core frequency to an 1800 MHz core frequency is <= 200 MHz.
2. Transition the processor core frequency from 1800 MHz to 2000 MHz (VCO frequency change is <= 200 MHz according to Table 44).

***Note:*** *To simplify this example, the voltage transitions and isochronous relief times are not described in this example.*

### Example 2:

Given a processor with a minimum core frequency of 1400 MHz and a maximum core frequency of 3400 MHz, to transition from 1400 MHz to 3400 MHz, the processor driver will (in the order specified):

1. Transition the processor from the core frequency of 1400 MHz to a core frequency of 3000 MHz. 3000 MHz is a portal frequency to and from 1400 MHz as defined in Table 43, because the VCO frequency change from 1400 MHz core frequency to 3000 MHz core frequency is <= 200 MHz.
2. Transition the processor core frequency from 3000 MHz to 3200 MHz (VCO frequency change is <= 200 MHz per Table 44).
3. Transition the processor core frequency from 3200 MHz to 3400 MHz (VCO frequency change is <= 200 MHz per Table 44).

***Note:*** *Note: to simplify this example, the voltage transitions and isochronous relief times are not described in this example.*

### Table 43. Low FID Frequency Table (< 1600 MHz)

| FID[5:0] | Minimum Core Frequency MHz | VCO Frequency MHz | Portal Core Frequencies in Table 44 |
|---|---|---|---|
| 000000b | 800 | 1600 | 1600, 1800 |
| 000010b | 1000 | 2000 | 1800, 2000, 2200 |
| 000100b | 1200 | 2400 | 2200, 2400, 2600 |
| 000110b | 1400 | 2800 | 2600, 2800, 3000 |

**Table 44. High FID Frequency Table (>= 1600 MHz)**

| FID[5:0] | Core Frequency MHz | VCO Frequency MHz |
|---|---|---|
| 001000b | 1600 | 1600 |
| 001010b | 1800 | 1800 |
| 001100b | 2000 | 2000 |
| 001110b | 2200 | 2200 |
| 010000b | 2400 | 2400 |
| 010010b | 2600 | 2600 |
| 010100b | 2800 | 2800 |
| 010110b | 3000 | 3000 |
| 011000b | 3200 | 3200 |
| 011010b | 3400 | 3400 |
| 011100b | 3600 | 3600 |
| 011110b | 3800 | 3800 |
| 100000b | 4000 | 4000 |
| 100010b | 4200 | 4200 |
| 100100b | 4400 | 4400 |
| 100110b | 4600 | 4600 |
| 101000b | 4800 | 4800 |
| 101010b | 5000 | 5000 |

### 9.5.5.2 P-state Transition Algorithm

The P-state transition algorithm has three phases. During phase 1 the processor voltage is transitioned to the level required to support frequency transitions. During phase 2 the processor frequency is transitioned to frequency associated with the OS-requested P-state. During phase 3 the processor voltage is transitioned to the voltage associated with the OS-requested P-state.

Figure 4 shows the high-level P-state transition flow. Figure 5 is a high-level timing diagram depicting voltage and frequency stepping associated with each phase of a P-state transition. Figure 6 shows the core voltage transition flow for phase 1 of a P-state transition. Figure 7 shows the core frequency transition flow for phase 2 of a P-state transition. Figure 8 shows the core voltage transition flow for phase 3 of a P-state transition.

The algorithm shown in Figure 4 through Figure 8 describes the 3-phases of a P-state transition in the context of Windows XP and the associated processor driver, but the algorithm also applies to the AMD PowerNow! driver and any other software that performs P-state transitions (such as a debug tool).

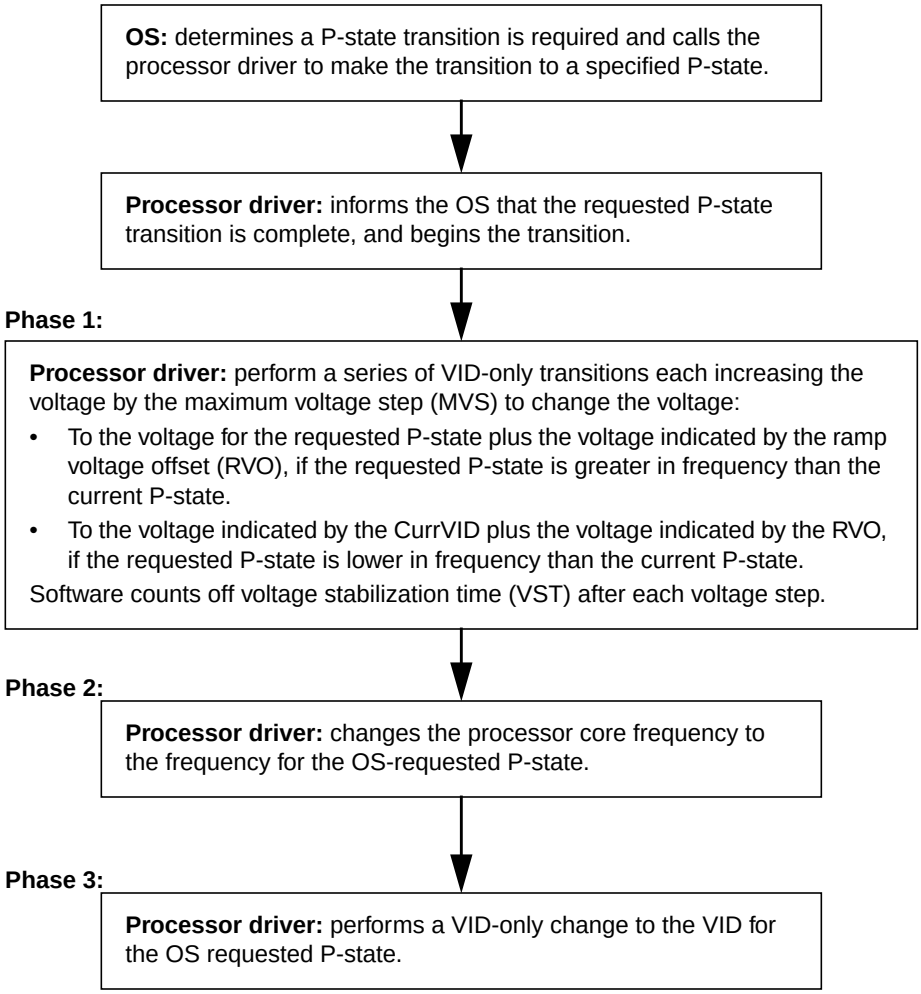



**Figure 4. High-Level P-state Transition Flow**

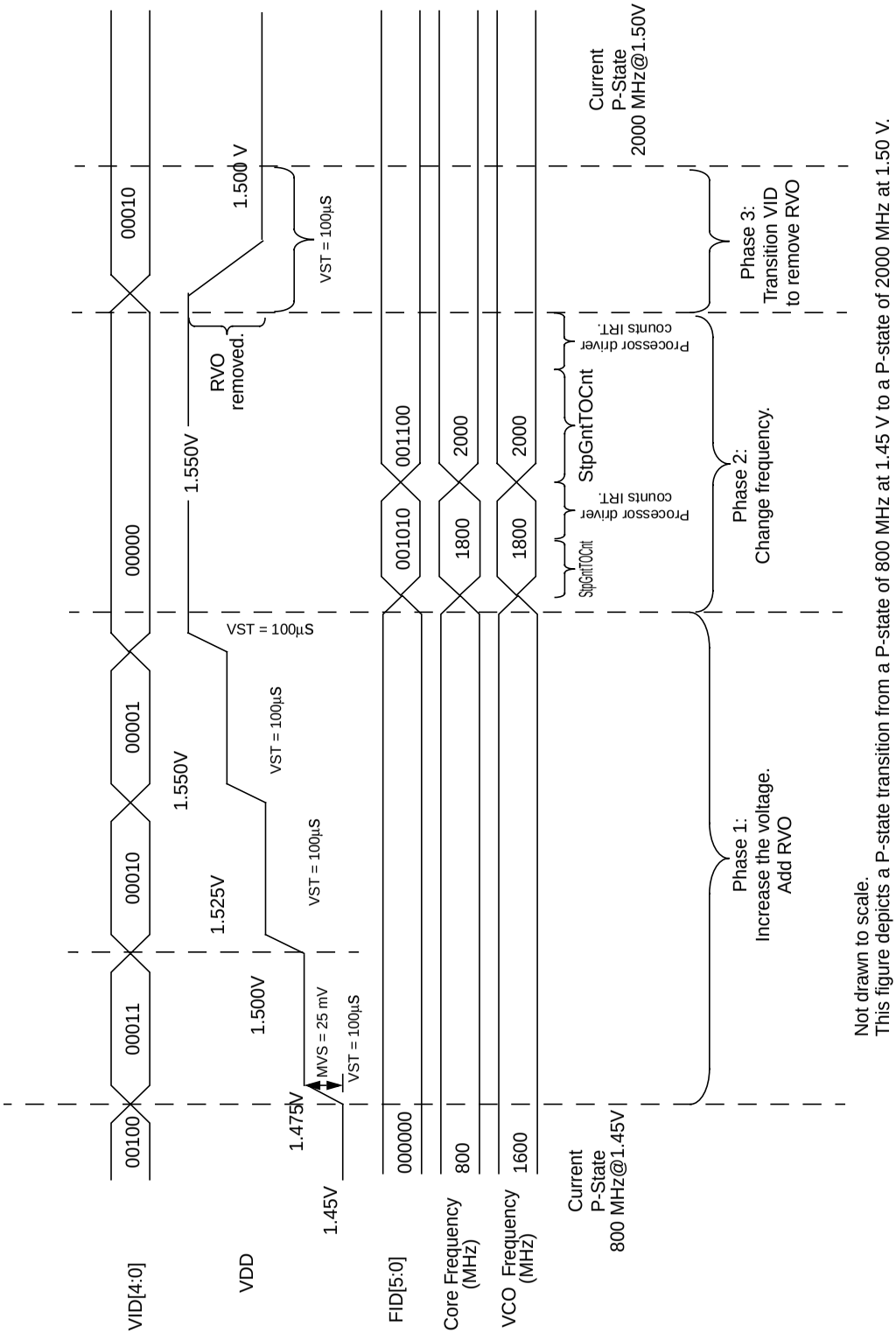



**Figure 5. Example P-State Transition Timing Diagram**

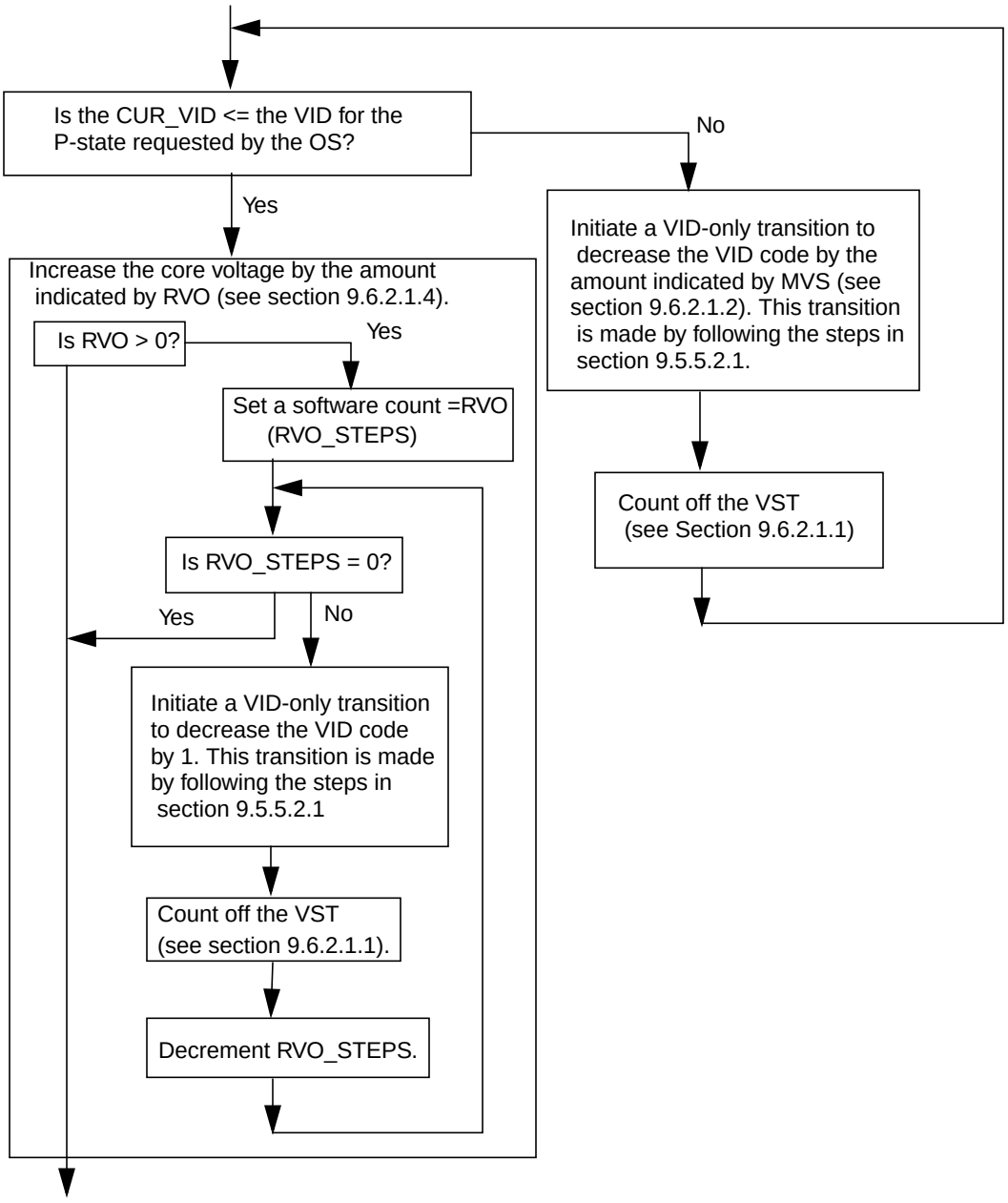



**Figure 6. Phase 1: Core Voltage Transition Flow**

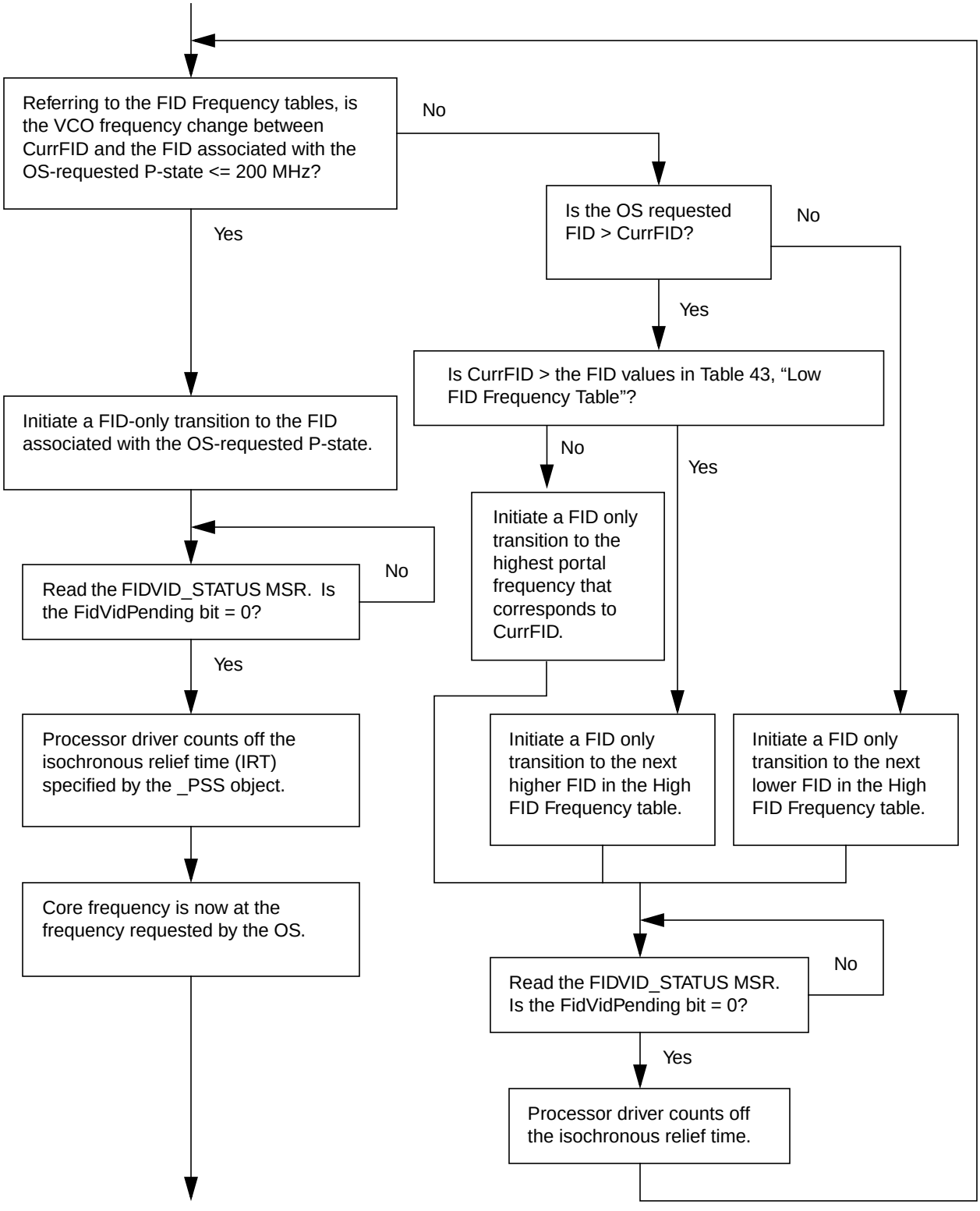



**Figure 7. Phase 2: Core Frequency Transition Flow**

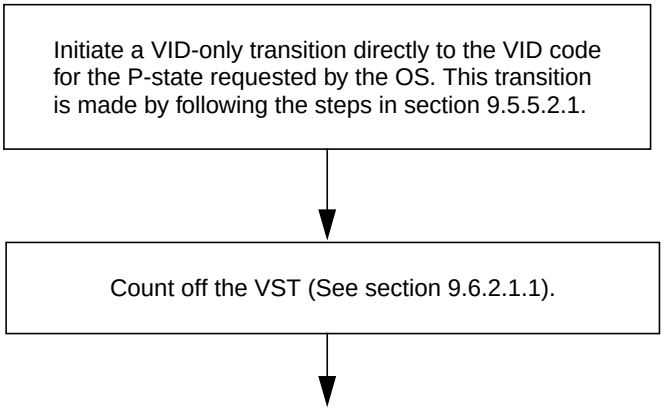



**Figure 8. Phase 3: Core Voltage Transition Flow**

### *9.5.5.2.1 Changing the VID*

***Note:*** *Software must hold the FID constant when changing the VID.*

To change the processor voltage:

1. Write the following values to FIDVID_CTL (MSR C001_0041h):
   - NewVID field (bits 12–8) with the VID code associated with the target voltage
   - NewFID field (bits 5–0) with the CurrFID value indicated in the FIDVID_STATUS MSR
   - StpGntTOCnt field (bits 51–32) with 1h, which corresponds to 5 nsec. For Revision B processors, this setting results in voltage transitions causing the minimum possible memory access latency. This field has no effect on VID changes for Revision C AMD Athlon™ 64 and AMD Opteron™ processors, since the processor drives the VID[4:0] in response to the MSR write without the processor first being placed into the Stop Grant state.
   - InitFidVid bit (bit 16) set to 1. Setting this bit initiates the VID change.
   - Clear all other bits to 0.
2. Loop on reading the FidVidPending bit (bit 31) of FIDVID_STATUS (MSR C001_0042h) until the bit returns 0. The FidVidPending bit stays set to 1 until the new VID code has been driven to the voltage regulator.

### *9.5.5.2.2 Changing the FID*

The AMD Athlon™ 64 and AMD Opteron™ processors support direct FID transitions only between FIDs that represent a VCO frequency change less then or equal to 200 MHz. To change between FIDs that represent a VCO frequency change greater than 200 MHz, software must make multiple transitions each less than or equal to a 200 MHz change in VCO frequency. For information about allowable FID transitions, see Tables 43 and 44 on page 224.

***Note:*** *Software must hold the VID constant when changing the FID.*

To change the core frequency:

1. Write the following values to FIDVID_CTL (MSR C001_0041h):
   - NewVID field (bits 12–8) with the CurrVID value reported in the FIDVID_STATUS MSR.
   - NewFID field (bits 5–0) with the FID code associated with the target frequency.
   - StpGntTOCnt field (bits 51–32) with a value corresponding to the processor PLL lock time. The PLL lock time is specified by the processor data sheet and refers to all components of PLL lock including frequency lock, phase lock, and settling time. PLL lock time is communicated to the processor driver through the PLL_LOCK_TIME field of the _PSS object. PLL lock time is in microseconds. To translate the PLL lock time to an StpGntTOCnt value, multiply PLL_LOCK_TIME by 1000 to get nanoseconds, then divide by 5 which is the clock period of the counter in ns. Therefore, StpGntTOCnt value = PLL_LOCK_TIME ∗1000/5. For example, a PLL lock time of 2 µs results in an StpGntTOCnt value of 400 decimal and 190h.
   - InitFidVid bit (bit 16) set to 1. Setting this bit initiates the P-state transition.
   - Clear all other bits to 0.
2. Loop on reading the FidVidPending bit (bit 31) of FIDVID_STATUS (MSR C001_0042h) until the bit returns 0. The FidVidPending bit stays set to 1 until the new FID code is in effect.

Figure 7 on page 228 illustrates the transition flow for core frequencies, including the requirement to wait the isochronous relief time between FID steps.

# 9.6 ACPI 2.0 Processor P-State Objects

For systems that support P-state transitions, the BIOS must provide the following ACPI 2.0 P-state objects to support operating systems that provide native support for processor P-state transitions. These subsections discuss the specific implementation of these objects.

## 9.6.1 _PCT (Performance Control)

The _PCT object is generically described in section 8.3.3.1 of the ACPI 2.0 specification. The _PCT, _PSS, and _PPC objects (defined in the next sections) are all placed after the Processor object in the _PR name space for operating systems that have native support for processor P-states. This section provides BIOS specifics regarding the use of _PCT.

The _PCT object specifies the control register and status register used to initiate and verify P-state transitions. The processor's P-state transition protocol is abstracted from the operating system by the processor driver, so this object does not declare the MSRs used for FID_Change protocol, but rather declares these two registers to be *functional fixed hardware*. Functional fixed hardware means that the mechanism is specific to the processor, and the processor driver knows how to initiate and verify P-state transitions because the processor driver is written by the processor vendor or written based upon documentation supplied to the operating system vendor by the processor vendor.

The following is a sample _PCT object. Sections referenced are in the ACPI 2.0 specification and ACPI 2.0 Errata Document, revision 1.4.

```
Name (_PCT, Package (2)         // Performance Control Object
        {
//            ResourceTemplate () {Register(FFixedHW, 0, 0, 0)}   // Control
         Buffer () {
                0x82,            // B0-Generic Register Descriptor (Sec. 6.4.3.7)
                 0xC,0,           // B1:2-length (from _ASI through _ADR fields)
                 0x7F,            // B3-Address space ID, _ASI, SystemIO
                 0,               // B4-Register Bit Width, _RBW
                 0,               // B5-Register Bit Offset, _RBO
                 0,               // B6-Reserved. Must be 0.
                 0,0,0,0,0,0,0,0, // B7:14-register address, _ADR (64bits)
                 0x79,0},         // B15:16-End Tag (Section 6.4.2.8)

//            ResourceTemplate () {Register(FFixedHW, 0, 0, 0)}      // Status
         Buffer () {
                 0x82,            // B0-Generic Register Descriptor (Section
6.4.3.7)
                 0xC,0,           // B1:2-length (2 bytes)
                 0x7F,            // B3-Address space ID, _ASI, SystemIO
                 0,               // B4-Register Bit Width, _RBW
                 0,               // B5-Register Bit Offset, _RBO
                 0,               // B6-Reserved. Must be 0.
                 0,0,0,0,0,0,0,0, // B7:14-register address, _ADR (64bits)
                 0x79,0},         // B15:16-End Tag (Section 6.4.2.8)
     }) // End of _PCT object
```

### 9.6.2 _PSS (Performance-Supported States)

The _PSS object is generically described in Section 8.3.3.2 of the ACPI 2.0 specification. This object follows the _PCT object in the _PR name space in support of operating systems with native support for processor P-states. This section provides the BIOS specifics regarding use of _PSS.

The _PSS object specifies the performance states that are supported by the system.

Based upon the maximum voltage and frequency supported by the processor and system requirements, the BIOS will build the _PSS object to specify the performance states supported by the system.

The BIOS creates the _PSS object based on the table of valid P-states in the processor data sheet. The BIOS is required to maintain a set of Performance State Tables (PSTs) based on the processor data sheet for all processors capable of P-state transitions. Only the PST entries that correspond to the processor found in the specific system by the BIOS during the POST routine are used by the BIOS to create the _PSS object for that system. The _PSS object could have as many performance states as the processor data sheet indicates are supported. For example, the _PSS object could provide performance state definitions P0–P2:

```
P0: 2400 MHz at 1.20 V
P1: 1600 MHz at 1.10 V
P2:  800 MHz at 1.00 V
```

The generic _PSS object description has the following format:

```
Name (_PSS, Package()
{          // Field Name Field Type
Package () // Performance State 0 Definition – P0
{
CoreFreq,            // DWordConst
Power,               // DWordConst
TransitionLatency,   // DWordConst
BusMasterLatency,    // DWordConst
Control,             // DWordConst
Status               // DWordConst
},
Package ()           // Performance State n Definition – Pn
{
CoreFreq,            // DWordConst
Power,               // DWordConst
TransitionLatency,   // DWordConst
BusMasterLatency,    // DWordConst
Control,             // DWordConst
Status               // DWordConst
}
} )                  // End of _PSS object
```

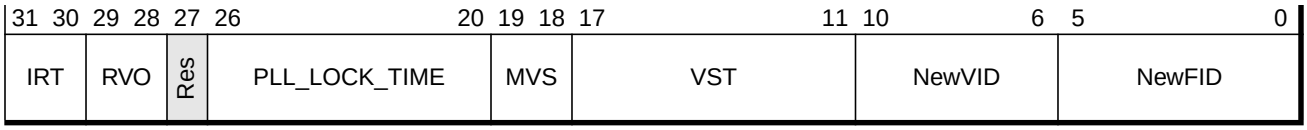
Each performance state entry contains six data fields as follows:

- ***CoreFreq*** is the core CPU operating frequency (in MHz).
- ***Power*** is the typical power dissipation (in milliWatts). The BIOS must fill out this field for the maximum performance state as specified in the processor data sheet. Estimates for power consumption for lower P-states are acceptable.
- ***TransitionLatency*** is the worst-case latency, in microseconds, that the CPU is unavailable during a transition from any performance state to this performance state. During a P-state transition, the CPU is unavailable for no more than 6 µs for each frequency step. While the total P-state transition could take up to 2 ms, the operating system is not blocked during the transition. Therefore the recommended value for this field is 100 µs.
- ***BusMasterLatency*** is the worst-case latency, in microseconds, that Bus Masters are prevented from accessing memory during a transition from any performance state to this performance state. This value is estimated at 6 µs.

- ***Control*** is the value to be written to the _PSS Control Field in order to initiate a transition to the performance state. The BIOS must fill this out as described on page 233.
- **Status** is the value that the processor driver can compare to a value read from the _PSS Status Field to ensure that the transition to the performance state was successful. The BIOS must fill this out as described on page 236.

### 9.6.2.1 _PSS Control Field

The control field of each performance state definition within the _PSS object contains the information that the processor driver uses. This includes NewVID and NewFID values that must be written into the FIDVID_CTL register (MSR C001_0041) to initiate transitions between P-states. _PSS Control field has the following format:




| Bits | Field | Description |
|---|---|---|
| 31–30 | IRT | Isochronous Relief Time |
| 29–28 | RVO | Ramp Voltage Offset |
| 27 | Reserved | Must be declared as zero. |
| 26–20 | PLL_LOCK_TIME | Phase-Locked Loop Time |
| 19–18 | MVS | Maximum Voltage Step |
| 17–11 | VST | Voltage Stabilization Time |
| 10–6 | NewVID | The VID value associated with the OS-requested P-state |
| 5–0 | NewFID | The FID value associated with the OS-requested P-state |

The following sections describe the sub-fields of the _PSS control field.

#### *9.6.2.1.1 Voltage Stabilization Time*

The Voltage Stabilization Time (VST) defines the number of microseconds (in 20 µs increments) that are required for the voltage to increase by the amount specified by the MVS field when the VID outputs of the processor transition. The processor driver counts VST between VID steps that increase the processor's core voltage, not between steps that decrease it. Table 45 shows the actual stabilization time for several VST values.

**Table 45. Sample VST Values**

| VST (Bits 17–11) | Stabilization Time |
|---|---|
| 7'h00 | 0 µs |
| 7'h01 | 20 µs |
| ... | |

**Table 45. Sample VST Values**

| VST (Bits 17–11) | Stabilization Time |
|---|---|
| 7'h05 | 100 µs (BIOS default. This is the value used by all platforms.) |
| … | |
| 7'h64 | 2000 µs |
| … | |
| 7'h7e | 2520 µs |
| 7'h7f | 2540 µs |

#### *9.6.2.1.2 Maximum Voltage Step*

The Maximum Voltage Step (MVS) defines the maximum voltage increment the processor driver can use when changing from a lower voltage to a higher voltage. For the processor driver, when increasing core voltage, the next VID = CurrVID – $2^{MVS}$. Table 46 on page 234 shows the values and increments for this field.

For example, if the voltage is being increased, and

- the current VID is 00100 (1.45 V),
- the OS-requested target VID is 00000 (1.55 V),
- and MVS is 0 (which selects 25mV voltage steps).

Then, the processor driver calculates the first VID to which to transition as follows:

Next_VID = CurrVID - $2^{MVS}$ = 00100 - $2^0$ = 00100 - 1 = 00011 = 1.475 V

Next the processor driver transitions the VID to select 1.475 V.

After the VST is counted off, the processor driver calculates the next VID to which to transtion as follows:

Next_VID = CurrVID - $2^{MVS}$ = 00011 - $2^0$ = 00010 = 1.500 V

This process iterates until the CurrVID is equal to the OS-requested target VID minus the VID associated with the ramp voltage offset. After each VID transition the processor driver counts off the voltage stabilization time before calculating the Next_VID.

**Table 46. MVS Values**

| MVS (Bits 19–18) | Increment |
|---|---|
| 00 | 25 mV (BIOS default. This is the required value that must be used.) |
| 01 | 50 mV (Reserved for future use) |
| 10 | 100 mV (Reserved for future use) |
| 11 | 200 mV (Reserved for future use) |

#### 9.6.2.1.3 PLL Lock Time

The 7-bit binary PLL_LOCK_TIME value defines the time required for the processor PLLs to lock in microseconds. The PLL_LOCK_TIME value is specified in the processor data sheet.

#### 9.6.2.1.4 Ramp Voltage Offset

The Ramp Voltage Offset (RVO) is the offset voltage applied to the nominal voltage when performing P-state transitions. The RVO is specified in the processor data sheet. Table 47 lists the RVO values and their corresponding offset voltages.

**Table 47. RVO Values**

| RVO (Bits 29–28) | Offset Voltage | Description |
|---|---|---|
| 00 | 0 mV | Target P-state VID needs no adjustment for RVO. |
| 01 | 25 mV | Decrement the target P-state VID by 1 to add the RVO. |
| 10 | 50 mV | Decrement the target P-state VID by 2 to add the RVO. (BIOS default. This is the required value.) |
| 11 | 75 mV | Decrement the target P-state VID by 3 to add the RVO. |

For the processor driver, RVO is in VID increments. To increase the processor voltage the processor driver must subtract the RVO field from the VID. The MVS field dictates the increments in which RVO can be subtracted from VID.

For example, consider the case when the VID is 00100b (1.45 V), RVO is 10b (50 mV), MVS is 00b (25 mV), and VST is 7'h05 (100 µs). Based upon these parameters, the voltage must be increased by 50 mV in two steps of 25 mV with a 100 µs delay after each VID change. The steps to apply the RVO are:

1. The VID code is reduced to 00011b. This transitions the voltage from 1.45 V to 1.475 V.
2. The processor driver waits 100 µs as specified by VST.
3. The VID code is reduced to 00010b. This transitions the voltage from 1.475 V to 1.5 V.
4. The processor driver waits 100 µs as specified by VST.

Table 48 lists the VID codes and their corresponding voltages for the AMD Athlon™ 64 and AMD Opteron™ processors.

**Table 48. VID Code Voltages**

| VID Code (Bits 4–0) | Voltage | VID Code (Bits 4–0) | Voltage |
|---|---|---|---|
| 00000 | 1.550 V | 10000 | 1.150 V |
| 00001 | 1.525 V | 10001 | 1.125 V |
| 00010 | 1.500 V | 10010 | 1.100 V |

**Table 48. VID Code Voltages (Continued)**

| VID Code (Bits 4–0) | Voltage | VID Code (Bits 4–0) | Voltage |
|---|---|---|---|
| 00011 | 1.475 V | 10011 | 1.075 V |
| 00100 | 1.450 V | 10100 | 1.050 V |
| 00101 | 1.425 V | 10101 | 1.025 V |
| 00110 | 1.400 V | 10110 | 1.000 V |
| 00111 | 1.375 V | 10111 | 0.975 V |
| 01000 | 1.350 V | 11000 | 0.950 V |
| 01001 | 1.325 V | 11001 | 0.925 V |
| 01010 | 1.300 V | 11010 | 0.900 V |
| 01011 | 1.275 V | 11011 | 0.875 V |
| 01100 | 1.250 V | 11100 | 0.850 V |
| 01101 | 1.225 V | 11101 | 0.825 V |
| 01110 | 1.200 V | 11110 | 0.800 V |
| 01111 | 1.175 V | 11111 | Off |

#### *9.6.2.1.5 Isochronous Relief Time*

The Isochronous Relief Time (IRT) is the amount of time the processor driver must count after each FID change step in a given P-state transition. During each FID change step, system memory is inaccessible to bus masters. While the processor driver is counting IRT between FID change steps, bus masters have access to system memory. Table 49 lists the IRT values and their corresponding times.

**Table 49. IRT Values**

| IRT (Bits 31–30) | Time |
|---|---|
| 00 | 10 µs |
| 01 | 20 µs |
| 10 | 40 µs |
| 11 | 80 µs (BIOS default) |

### 9.6.2.2 _PSS Status Field

The status field of each performance state definition within the _PSS object contains the information required by the processor driver to verify that the transition has completed based upon a read of the FIDVID_STATUS MSR. The _PSS Status field is 32 bits, and the BIOS must map the FIDVID_STATUS MSR CurrVID and CurrFID information to the _PSS status field as shown in Table 50 on page 237.

**Table 50. _PSS Status Field**

| Bit | Name | Description |
|---|---|---|
| 5–0 | CurrFID | The FID value associated with the OS-requested P-state |
| 10–6 | CurrVID | The VID value associated with the OS-requested P-state |
| 31–11 | Reserved | These bits must be 0 and are ignored by the processor driver. |

The CurrFID and CurrVID fields of the FIDVID_STATUS MSR are valid only when the FidVidPending bit is 0.

## 9.6.3 _PPC (Performance Present Capabilities)

The _PPC object is used to limit the maximum P-state that the operating system can use at any given time. This object follows the _PSS object in the _PR name space for the operating systems that support the ACPI 2.0 processor P-state objects.

_PPC is generically described in Section 8.3.3.3 of the ACPI 2.0 specification.

This section provides BIOS specifics regarding use of _PPC.

If the OEM and BIOS vendor do not implement a _PPC that limits the maximum P-state under certain conditions, then the BIOS must always return a value of 0 for the _PPC object. When a value of 0 is returned for the _PPC object, all P-states specified by the _PSS object are available.

Operating systems with native support for processor P-state transitions and ACPI Thermal zones make use of reduced P-states for passive cooling before making use of "throttling" with the stop grant state. This is the case even when the _PPC object returns a value of 0.

### 9.6.3.1 Using _PPC to Limit the Maximum Processor P-State When battery Powered

Processor performance exceeds the demands of most typical notebook PC applications. However, there are basically idle scenarios where applications run instructions that incorrectly cause the operating system to conclude that the processor is 100% utilized. This causes the operating system to force the processor to its maximum performance state, wasting power with no benefit to the user. Also, static power consumption of the processor increases when the processor is operating in a P-state above the minimum voltage level supported by the processor.

Therefore, it will increase the system's battery-powered runtime to limit the maximum performance state available when battery-powered.

### 9.6.3.2 Implementation Overview

An ACPI-compliant General-Purpose Event (GPE) input of the chipset (typically the I/O Hub) is dedicated to causing an SCI whenever the notebook power source changes. This input will be referred

to as AC available (ACAV). The system can control ACAV directly based upon the presence of an AC adapter, or the embedded controller (EC) in the system can control ACAV.

Whenever the AC adapter is inserted or removed from the system, the chipset logic associated with ACAV sets the status bit associated with ACAV, and if enabled causes an SCI to be asserted. Note: the control method associated with ACAV status could be implemented in the EC. A chipset GPE dedicated for AC adapter status is used for this description.

The BIOS provides the following ACPI ASL control methods and objects:

- _PPC (Performance Present Capabilities)
- _PSR (Power Source) is described in Section 11.3.1 of the ACPI 2.0 specification. _PSR returns:
  - 0x00000000 if the system is running on battery power.
  - 0x00000001 if the system is running from an AC adapter.
- _SB.AC is a BIOS-defined device object for the AC adapter.
- _L12 is the control method associated with GPE 12.

The flow associated with AC adapter insertion or removal is:

- AC adapter is inserted or removed
- The GPE Status bit associated with ACAV is asserted (for this example, GPE 12)
- An SCI (system control interrupt is issued)
- The OS/ACPI driver determines that GPE 12 was asserted and runs the associated control method (_L12)
- _L12
  - Issues a Notify(_PR.CPU0, 0x80) which forces the OS to re-evaluate the _PPC object.
    - _PPC reads the state of the ACAV pin and returns:

      0 if the system is AC-powered. All P-states are available.

      n if the system is battery-powered, where n is the highest P-state is supported when battery-powered. Only P-states n and lower are available. For example, if a processor supports 3 P-states:
      P0 = 2400 MHz at 1.2 V
      P1 = 1600 MHz at 1.1 V
      P2 = 800 MHz at 1.0 V

      Then n = 2 will limit the maximum P-state to 800 MHz at 1.0 V when the system is battery-powered.
    - OS takes into account the available P-states and if necessary performs a P-state transition.
  - Issues a Notify(_SB.AC, 0x80) where _SB.AC is the AC adapter device object.

– OS runs _PSR to determine if the current power source is the AC adapter. PSR reads the state of the ACAV pin and returns:

1 if the system is AC-powered.

0 if the system is battery-powered.

- Normal operation continues.

### 9.6.4 PSTATE_CNT

PSTATE_CNT is an entry in the Fixed ACPI Description Table (FADT). A PSTATE_CNT entry of 0 informs the operating system that the BIOS in System Management Mode (SMM) does not perform P-state transitions.

The BIOS must report a value of 0 in the PSTATE_CNT field of the FADT. The only BIOS-controlled P-state transition, if any, must be performed near the beginning of the POST routine before control is passed to the operating system. All subsequent transitions are made by system software not the BIOS. System software is either the AMD PowerNow! technology software (for Microsoft® operating systems prior to the Windows® XP operating system) or the operating system and associated processor driver (for the Windows XP operating system and subsequent Microsoft operating systems).

***Note:*** *SMM is not used to perform P-state transitions.*

### 9.6.5 CST_CNT

CST_CNT is an entry in the FADT. If non-zero, this field contains the value the operating system writes to the SMI_CMD register to indicate operating system support for the _CST object and C States Changed notification.

The BIOS must report a value of 0 in the CST_CNT field of the FADT. AMD platforms only support ACPI 1.0b processor power state functionality. Processor power states are referred to as "C" states by the ACPI specifications.

## 9.7 BIOS Support for AMD PowerNow!™ Software with Legacy Operating Systems

If BIOS determines it has a processor and system that support P-state transitions, it provides a PSB that is comprised of a header and the processor specific Performance State Table (PST) that matches the processor in the system.

If the BIOS cannot determine that the system and processor support P-state transitions as defined in 9.5.1 on page 219, then the BIOS must not provide a PSB. Changing the PSB Signature field to all 0s is an effective way of removing the PSB.

Table 51 defines the PSB and PST structures.

**Table 51. Performance State Block Structure**

| Name | Length (Bytes) | Field Purpose |
|---|---|---|
| **PSB Header** | | |
| Signature | 10 | Start of the PSB - Always set this field to "AMDK7PNOW!" |
| TableVersion | 1 | The version of the table as defined by AMD. Set to 14h for version 1.4. If AMD PowerNow! software does not recognize the TableVersion, it will not attempt to make P-state transitions. |
| Flags | 1 | All bits are reserved and written to zero. |
| VST | 2 | Voltage Stabilization Time. The amount of time required for the processor's core voltage regulator to increase the processor's core voltage the increment specified by MVS. VST is specified in 20 µs increments. The default is 05h (100 µs). All other values are reserved. |
| Reserved1 | 1 | Bits 1–0: Ramp Voltage Offset (RVO) has the same definition as for the _PSS ACPI object.<br>Bits 3–2: Isochronous Relief Time (IRT) has the same definition as for the _PSS ACPI object.<br>Bits 5–4: Maximum Voltage Step (MVS) has the same definition as for the _PSS ACPI object.<br>Bits 7–6: Number of available P-states when the system is battery powered has the following definition:<br>00b = all P-states are available<br>01b = only the minimum P-state is available<br>10b = 2 lowest P-states are available<br>11b = 3 lowest P-states are available. |
| NumPST | 1 | The total number of PSTs in the PSB. This value must be set to 01h. |
| **PST Header** | | |
| CPUID | 4 | CPUID (EAX of CPUID extended function 8000_0001h) of the processor that this PST belongs to. This provides processor family, extended family, model, extended model, and stepping (revision) fields. |
| PLL Lock Time | 1 | Processor PLL Lock time in microseconds.The PLL Lock time is specified in the processor data sheet addendum. For example, PLL_LOCK_TIME of 2 µs is represented as 00000010b. |
| MaxFID | 1 | MaxFID[5:0] as reported by the MaxFID field of the FIDVID_STATUS MSR. |
| MaxVID | 1 | MaxVID[4:0] as reported by the MaxVID field of the FIDVID_STATUS MSR. |
| NumPStates | 1 | The number of FID, VID combinations in the PST. This field must be >= 1h. |
| FID | 1 | FID[5:0] code corresponding to the frequency of the lowest performance state that the processor supports. This is defined in the processor data sheet. |
| VID | 1 | VID[4:0] code corresponding to the voltage of the lowest performance state that the processor supports. This is defined in the processor data sheet. |

**Table 51. Performance State Block Structure (Continued)**

| Name | Length (Bytes) | Field Purpose |
|---|---|---|
| ---------- | --------- | FID, VID pairs are concatenated here so that the total number of pairs is equal to the NumPstates field. Each subsequent FID, VID pair is appended in ascending order (i.e., the last FID,VID pair corresponds to the maximum performances state supported by the processor). |

For processors with a family number of less than Fh, refer to BIOS Requirements for AMD PowerNow! Technology Application Note, order# 25264.

# 9.8 System Configuration for Power Management

## 9.8.1 Chipset Configuration for Power Management

Chipset configuration is covered in chipset related documentation. The BIOS is required to:

- Ensure that the SMAF code to STPCLK and Stop Grant mapping for all components based on HyperTransport technology is configured as described in Table 42 on page 217.
- For single processor systems, the HyperTransport links are tri-stated in response to LDTSTOP_L assertion.
- Chipset power management features are appropriately enabled based upon system class.
- LSTSTOP_L assertion during FID_Change:
  - For Revision B processors is at least 4µs but less than 10 µs for UP systems.
  - For Revision C and subsequent processors is at least 2µs but less than 6µs. Note: for UMA chipsets 2µs is ideal and may be required in some cases.
- LSTSTOP_L assertion during FID_Change is 10 µs for multiprocessor systems.

## 9.8.2 Processor Configuration for Power Management

BIOS configures the processor to enable power management during the POST routine. Table 52 on page 242 lists processor memory system configuration register settings that the BIOS must initialize.

## Table 52. Configuration Register Settings for Power Management

| Functionality | Register | Mobile Setting | UP Desktop Setting | Multiprocessor Setting |
|---|---|---|---|---|
| Do not enable HyperTransport™ links to be tri-stated in response to LDTSTOP_L assertion | LDT0 Link Control (function 0: offset 84h[13]) (LdtStopTriEn)<br>LDT1 Link Control (function 0: offset A4h[13]) (LdtStopTriEn)<br>LDT2 Link Control (function 0: offset C4h[13]) (LdtStopTriEn) | 0b | 0b | 0b |
| Map STPCLK actions to SMAF codes:<br>SMAF 0 = C2<br>SMAF 1 = C3<br>SMAF 2 = VID/FID Change<br>SMAF 3 = S1 | Power Management Control Low (function 3: offset 80h) | 6307_6361h | 2307_0000h | Revision B: 2117_0000h<br>Revision C: 2307_0000h |
| Map STPCLK actions to SMAF codes:<br>SMAF 4 = S3<br>SMAF 5 = Throttling<br>SMAF 6 = S4/S5<br>SMAF 7 is not used for STPCLK, the corresponding field is used for HALT. | Power Management Control High (function 3: offset 84h) | 6113_4113h | 2113_2113h | Revision B: 0013_0013h<br>Revision C[1]: 0013_2113h |
| BCLK and LCLK PLL frequency lock | Clock Power/Timing Low (function 3: offset D4h) | Revision B: 000D_0001h<br>Revision C with unbuffered memory: 000D_0701h<br>Revision C with registered DIMMs: 04E2_0707h | | |
| Sets "Ramp VID" (0 mV) and Voltage Regulator Stabilization time (0 s). The processor driver performs this function, not hardware. | Clock Power/Timing High (function 3:offset D8h) | 0000_0000h | 0000_0000h | 0000_0000h |

***Notes:***

1. *Optionally, a value of 0113_2113h can be used. This will significantly reduce processor power during halt and may reduce system performance.*

#### 9.8.2.1 Additional BIOS Support Requirements for S3

During the S3 state, power is removed from the processor's CPU core, host bridge, and memory controller. It is required that the BIOS restore the state of the processor's memory controller upon resume from S3 prior to having the memory controller bring system memory out of self-refresh mode. The system must provide non volatile storage for the memory controller configuration registers during the S3 state. For UP and DP systems, this storage will be provided in the chipset. There are two opportunities for the BIOS to store the memory controller configuration to this nonvolatile memory.

- During POST as the memory controller is being initially configured—this is preferred, since it does not require SMM support.
- In SMM just prior to entry to the S3 state—BIOS uses an SMI trap on writes to the ACPI PM1 control register to get control of the system just prior to entry to S3. After storing the memory controller configuration registers, I/O instruction restart is used to place the system into the S3 state. This is not preferred because it relies on SMM.

Processor Function 2 configuration space registers at offsets 40–98h must be stored in nonvolatile RAM for support of resume from S3.

#### 9.8.2.2 ACPI Tables/Objects

FADT entries for C-state and throttling support and throttling is listed in Table 53 on page 243. Some legacy operating systems will not use C3 if support for C2 is not declared.

**Table 53. FADT Table Entries**

| Field | Mobile | UP Desktop | Multiprocessor |
|---|---|---|---|
| PSTATE_CNT | 0 | 0 | 0 |
| CST_CNT | 0 | 0 | 0 |
| P_LVL2_LAT | > 100 if C2 is not supported<br>1 if C2 is supported | > 100 | > 100 |
| P_LVL3_LAT | > 1000 if C3 is not supported<br>10 if C3 is supported | > 1000 | > 1000 |
| DUTY_WIDTH | 0 if _PSV is not used<br>3 if _PSV is used | 0 if _PSV is not used<br>3 if _PSV is used | 0 |

# 10 Performance Monitoring

AMD Athlon™ 64 and AMD Opteron™ processors support the performance-monitoring features introduced in earlier processor implementations. These features allow the selection of events to be monitored, and include a set of corresponding counter registers that track the frequency of monitored events. Software tools can use these features to identify performance bottlenecks, such as sections of code that have high cache-miss rates or frequently mis-predicted branches. This information can then be used as a guide for improving or eliminating performance problems through software optimizations or hardware-design improvements.

For a general description of how to use these performance monitoring features, refer to the "Debug and Performance Resources" section in Volume 2: System Programming of the *AMD64 Architecture Programmers Manual*, order# 24593.

## 10.1 Performance Counters

The processor provides four 48-bit performance counters. Each counter can monitor a different event. The accuracy of the counters is not ensured. Some events are reserved. When a reserved event is selected, the results are undefined.

Performance counters are used to count specific processor events, such as data-cache misses, or the duration of events, such as the number of clocks it takes to return data from memory after a cache miss. During event counting, the processor increments the counter when it detects an occurrence of the event. During duration measurement, the processor counts the number of processor clocks it takes to complete an event. Each performance counter can be used to count one event, or measure the duration of one event at a time.

The processor supports four performance counters, PerfCtr*n*. The implementation of each counter supports 48 bits of resolution. This section shows the format of the PerfCtr*n* registers.

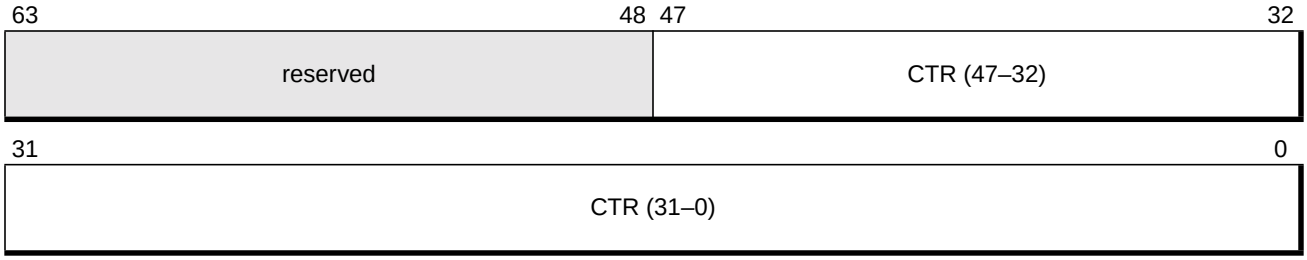
Only code executing at privilege level 0 can directly write the performance counters using the WRMSR instruction.The PerfCtr*n* registers are model-specific registers that can be read using a special *read performance-monitoring counter* instruction, RDPMC. The RDPMC instruction loads the contents of the PerfCtr*n* register specified by the ECX register, into the EDX register and the EAX register. The high 32 bits are loaded into EDX, and the low 32 bits are loaded into EAX. RDPMC can be executed only at CPL=0, unless system software enables use of the instruction at all privilege levels. RDPMC can be enabled for use at all privilege levels by setting CR4.PCE (the performance-monitor counter-enable bit) to 1. When CR4.PCE = 0 and CPL > 0, attempts to execute RDPMC result in a general-protection exception (#GP).

The performance counters can also be read and written by system software running at CPL=0 using the RDMSR and WRMSR instructions, respectively. Writing the performance counters can be useful if software wants to count a specific number of events, and then trigger an interrupt when that count is

reached. An interrupt can be triggered when a performance counter overflows. Software should use the WRMSR instruction to load the count as a two's-complement negative number into the performance counter. This causes the counter to overflow after counting the appropriate number of times.

The performance counters are not assured of producing identical measurements each time they are used to measure a particular instruction sequence, and they should not be used to take measurements of very small instruction sequences. The RDPMC instruction is not serializing, and it can be executed out-of-order with respect to other instructions around it. Even when bound by serializing instructions, the system environment at the time the instruction is executed can cause events to be counted before the counter value is loaded into EDX:EAX.

### PerfCtr0–PerfCtr3 Registers C001_0004h, C001_0005h, C001_0006h, C001_0007h




| Bit | Name | Function | R/W |
|---|---|---|---|
| 63–48 | Reserved | RAZ | R |
| 47–0 | CTR | Counter value | R |

## 10.2 Performance Event-Select Registers

Performance Event-Select registers (PerfEvtSel*i*) are used to specify the events counted by the performance counters, and to control other aspects of their operation. Each performance counter supported by the implementation has a corresponding event-select register that controls its operation. This section shows the format of the PerfEvtSel*n* registers.

The performance event-select registers can be read and written only by system software running at CPL=0 using the RDMSR and WRMSR instructions, respectively. Any attempt to read or write these registers at CPL>0 causes a general-protection exception to occur.

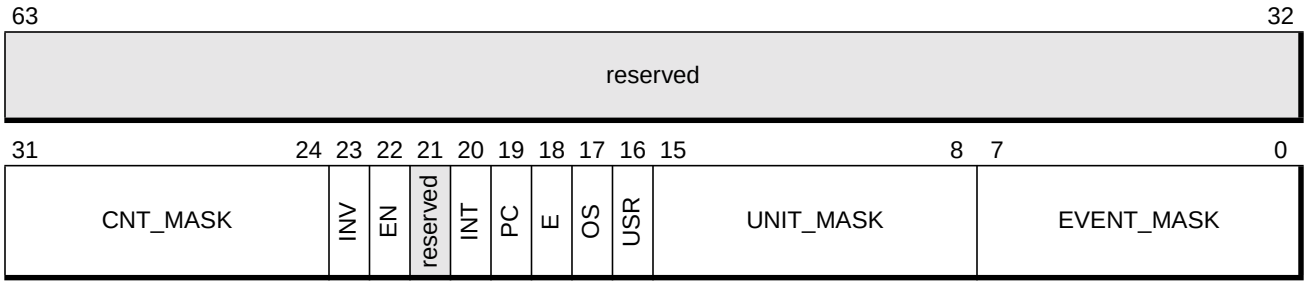
To accurately start counting with the write that enables the counter, disable the counter when changing the event and then enable the counter with a second MSR write.

The edge count mode will increment the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge will be falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.

The EVENT_MASK field specifies the event or event duration in a processor unit to be counted by the corresponding PerfCtr*i* register. Table 55 on page 248 shows the function unit encoding of EVENT_MASK[7:5]. The UNIT_MASK field is used to further specify or qualify a monitored event. Table 55 on page 248 shows the events and unit masks supported by the processor. The events that can be counted are implementation dependent. For more up-to-date information on the events in the latest processor revisions, refer to the *Revision Guide for the AMD Athlon™ 64 and AMD Opteron™ processors*, order# 25759.

### PerfEvtSel0–PerfEvtSel3 Registers

**C001_0000h, C001_0001h, C001_0002h, C001_0003h**




)

| Bit | Name | Function | R/W |
|---|---|---|---|
| 63–32 | reserved | RAZ | |
| 31–24 | CNT_MASK | Counter Mask | R/W |
| 23 | INV | Invert Counter Mask | R/W |
| 22 | EN | Enable Counter | R/W |
| 21 | reserved | SBZ | |
| 20 | INT | Enable APIC Interrupt | R/W |
| 19 | PC | Pin Control | R/W |
| 18 | E | Edge Detect | R/W |
| 17 | OS | Operating-System Mode | R/W |
| 16 | USR | User Mode | R/W |
| 15–8 | UNIT_MASK | Event Qualification | R/W |
| 7–0 | EVENT_MASK | Unit and Event Selection | R/W |

### Field Descriptions

**Event Selection (EVENT_MASK)**—Bits 7–0. EVENT_MASK[7:5] selects a processor functional unit. EVENT_MASK[4:0] selects an event within the processor unit.

**Event Qualification (UNIT_MASK)**—Bits 15–8. Each UNIT_MASK bit further specifies or qualifies the event specified by EVENT_MASK. All events selected by UNIT_MASK are simultaneously monitored.

**User Mode (USR)**—Bit 16. Count events in user mode (at CPL > 0).

**Operating-System Mode (OS)**—Bit 17. Count events in operating-system mode (at CPL = 0).

**Edge Detect (E)**—Bit 18. 1=Edge detection. 0=Level detection.

**Pin Control (PC)**—Bit 19. With a value of 1, toggles the performance monitor pin PM*i* when PerfCtr*i* register overflows. With a value of 0, toggles the performance monitor pin PM*i* each time PerfCtr*i* register is incremented.

**Enable APIC Interrupt (INT)**—Bit 20. Enables APIC interrupt when PerfCtr*i* register overflows.

**Enable Counter (EN)**—Bit 22. Enables performance monitor counter.

**Invert Counter Mask (INV)**—Bit 23. See CNT_MASK.

**Counter Mask (CNT_MASK)**—Bits 31–24. a) When CNT_MASK is 0, the corresponding PerfCtr*i* register is incremented by the number of events occurring in a clock cycle. Maximum number of events in one cycle is 3. b) When CNT_MASK values are from 1 to 3 and INV is 0, the corresponding PerfCtr*i* register is incremented by 1, if the number of events occurring in a clock cycle is greater than or equal to CNT_MASK. When CNT_MASK values are from 1 to 3 and INV is 1, the corresponding PerfCtr*i* register is incremented by 1, if the number of events occurring in a clock cycle is less than CNT_MASK. c) CNT_MASK values from 4 to 255 are reserved.

**Table 54. Processor Functional Unit Encoding**

| EVENT_MASK[7:5] | Description |
|---|---|
| 0 | Selects event from FP unit |
| 1 | Selects event from LS unit |
| 2 | Selects event from DC unit |
| 3 | Selects event from BU unit |
| 4 | Selects event from IC unit |
| 5 | Reserved |
| 6 | Selects event from FR unit |
| 7 | Selects event from NB unit |

**Table 55. Performance Monitor Events**

| EVENT_MASK [7:0] | Encoded EVENT_MASK [7:5] | UNIT_ MASK bit | Description |
|---|---|---|---|
| 00h | FP | | Dispatched FPU ops (Revision B and later revisions) |
| | | 0 | Add pipe ops excluding junk ops |
| | | 1 | Multiply pipe ops excluding junk ops |
| | | 2 | Store pipe ops excluding junk ops |
| | | 3 | Add pipe junk ops |
| | | 4 | Multiply pipe junk ops |

**Table 55. Performance Monitor Events (Continued)**

| EVENT_MASK [7:0] | Encoded EVENT_MASK [7:5] | UNIT_MASK bit | Description |
|---|---|---|---|
| | | 5 | Store pipe junk ops |
| | | 7–6 | Reserved |
| 01h | FP | | Cycles with no FPU ops retired (Revision B and later revisions) |
| 02h | FP | | Dispatched FPU ops that use the fast flag interface (Revision B and later revisions) |
| 03h to 1Fh | | | Reserved |
| 20h | LS | | Segment register load |
| | | 0 | ES |
| | | 1 | CS |
| | | 2 | SS |
| | | 3 | DS |
| | | 4 | FS |
| | | 5 | GS |
| | | 6 | HS |
| | | 7 | Reserved |
| 21h | LS | | Microarchitectural resync caused by self-modifying code |
| 22h | LS | | Microarchitectural resync caused by snoop |
| 23h | LS | | LS buffer 2 full |
| 24h | LS | | Locked operation (Revision B and earlier revisions) |
| | | 0 | Number of lock instructions executed (Revision C and later revisions) |
| | | 1 | Number of cycles spent in the lock request/grant stage (Revision C and later revisions) |
| | | 2 | Number of cycles a lock takes to complete once it is non-speculative and is the oldest load/store operation (non-speculative cycles in Ls2 entry 0) (Revision C and later revisions) |
| | | 7-3 | Reserved |
| 25h | LS | | Microarchitectural late cancel of an operation |
| 26h | LS | | Retired CFLUSH instructions |
| 27h | LS | | Retired CPUID instructions |
| 28h to 3Fh | | | Reserved |
| 40h | DC | | Access (includes microcode scratchpad accesses) |
| 41h | DC | | Miss |
| 42h | DC | | Refill from L2 |

**Table 55. Performance Monitor Events (Continued)**

| EVENT_MASK [7:0] | Encoded EVENT_MASK [7:5] | UNIT_MASK bit | Description |
|---|---|---|---|
| | | 0 | Invalid |
| | | 1 | Shared |
| | | 2 | Exclusive |
| | | 3 | Owner |
| | | 4 | Modified |
| | | 7–5 | Reserved |
| 43h | DC | | Refill from system |
| | | 0 | Invalid |
| | | 1 | Shared |
| | | 2 | Exclusive |
| | | 3 | Owner |
| | | 4 | Modified |
| | | 7–5 | Reserved |
| 44h | DC | | Copyback |
| | | 0 | Invalid |
| | | 1 | Shared |
| | | 2 | Exclusive |
| | | 3 | Owner |
| | | 4 | Modified |
| | | 7–5 | Reserved |
| 45h | DC | | L1 DTLB miss and L2 DTLB hit |
| 46h | DC | | L1 DTLB miss and L2 DTLB miss |
| 47h | DC | | Misaligned data reference |
| 48h | DC | | Microarchitectural late cancel of an access |
| 49h | DC | | Microarchitectural early cancel of an access |
| 4Ah | DC | | One bit ECC error recorded found by scrubber |
| | | 0 | Scrubber error |
| | | 1 | Piggyback scrubber errors |
| | | 7–2 | Reserved |
| 4Bh | DC | | Dispatched prefetch instructions |
| | | 0 | Load |
| | | 1 | Store |
| | | 2 | NTA |
| | | 3–7 | Reserved |
| 4Ch | DC | | DCACHE accesses by locks (Revision C and later revisions) |

**Table 55. Performance Monitor Events (Continued)**

| EVENT_MASK [7:0] | Encoded EVENT_MASK [7:5] | UNIT_MASK bit | Description |
|---|---|---|---|
| | | 0 | Number of dcache accesses by lock instructions (Revision C and later revisions) |
| | | 1 | Number of dcache misses by lock instructions (Revision C and later revisions) |
| | | 7-2 | Reserved |
| 4Dh to 5Fh | | | Reserved |
| 7Dh | BU | | Internal L2 request |
| | | 0 | IC fill |
| | | 1 | DC fill |
| | | 2 | TLB reload |
| | | 3 | Tag snoop request |
| | | 4 | Cancelled request |
| | | 7–5 | Reserved |
| 7Eh | BU | | Fill request that missed in L2 |
| | | 0 | IC fill |
| | | 1 | DC fill |
| | | 2 | TLB reload |
| | | 7–3 | Reserved |
| 7Fh | BU | | Fill into L2 |
| | | 0 | Dirty L2 victim |
| | | 1 | Victim from L1 |
| | | 7–2 | Reserved |
| 80h | IC | | Fetch |
| 81h | IC | | Miss |
| 82h | IC | | Refill from L2 |
| 83h | IC | | Refill from system |
| 84h | IC | | L1ITLB miss and L2ITLB hit |
| 85h | IC | | L1ITLB miss and L2ITLB miss |
| 86h | IC | | Microarchitectural resync caused by snoop |
| 87h | IC | | Instruction fetch stall |
| 88h | IC | | Return stack hit |
| 89h | IC | | Return stack overflow |
| 8Ah to BFh | | | Reserved |
| C0h | FR | | Retired x86 instructions including exceptions and interrupts |
| C1h | FR | | Retired uops |

**Table 55. Performance Monitor Events (Continued)**

| EVENT_MASK [7:0] | Encoded EVENT_MASK [7:5] | UNIT_MASK bit | Description |
|---|---|---|---|
| C2h | FR | | Retired branches including exceptions and interrupts |
| C3h | FR | | Retired branches mispredicted |
| C4h | FR | | Retired taken branches |
| C5h | FR | | Retired taken branches mispredicted |
| C6h | FR | | Retired far control transfers (always mispredicted) |
| C7h | FR | | Retired resyncs (non control transfer branches) |
| C8h | FR | | Retired near returns |
| C9h | FR | | Retired near returns mispredicted |
| CAh | FR | | Retired taken branches mispredicted only due to address miscompare |
| CBh | FR | | Retired FPU instructions (Revision B and later revisions) |
| | | 0 | x87 instructions |
| | | 1 | Combined MMX™ and 3DNow!™ instructions |
| | | 2 | Combined packed SSE and SSE2 instructions |
| | | 3 | Combined scalar SSE and SSE2 instructions |
| | | 4–7 | Reserved |
| CCh | FR | | Retired fastpath double op instructions (Revision B and later revisions) |
| | | 0 | With low op in position 0 |
| | | 1 | With low op in position 1 |
| | | 2 | With low op in position 2 |
| | | 3–7 | Reserved |
| CDh | FR | | Interrupts masked cycles (IF=0) |
| CEh | FR | | Interrupts masked while pending cycles (INTR while IF=0) |
| CFh | FR | | Taken hardware interrupts |
| D0h | FR | | Nothing to dispatch (decoder empty) |
| D1h | FR | | Dispatch stalls (D2h–DAh combined) |
| D2h | FR | | Dispatch stall from branch abort to retire |
| D3h | FR | | Dispatch stall for serialization |
| D4h | FR | | Dispatch stall for segment load |
| D5h | FR | | Dispatch stall when reorder buffer is full |
| D6h | FR | | Dispatch stall when reservation stations are full |
| D7h | FR | | Dispatch stall when FPU is full |
| D8h | FR | | Dispatch stall when LS is full |
| D9h | FR | | Dispatch stall when waiting for all to be quiet |

**Table 55. Performance Monitor Events (Continued)**

| EVENT_MASK [7:0] | Encoded EVENT_MASK [7:5] | UNIT_MASK bit | Description |
|---|---|---|---|
| DAh | FR | | Dispatch stall when far control transfer or resync branch is pending |
| DBh | FR | | FPU exceptions (Revision B and later revisions) |
| | | 0 | x87 reclass microfaults |
| | | 1 | SSE retype microfaults |
| | | 2 | SSE reclass microfaults |
| | | 3 | SSE and x87 microtraps |
| | | 4–7 | Reserved |
| DCh | FR | | Number of breakpoints for DR0 |
| DDh | FR | | Number of breakpoints for DR1 |
| DEh | FR | | Number of breakpoints for DR2 |
| DFh | FR | | Number of breakpoints for DR3 |
| E0h | NB | | Memory controller page access event |
| | | 0 | Page hit |
| | | 1 | Page miss |
| | | 2 | Page conflict |
| | | 7–3 | Reserved |
| E1h | NB | | Memory controller page table overflow |
| E2h | NB | | Memory controller DRAM command slots missed (in MemClks) |
| E3h | NB | | Memory controller turnaround |
| | | 0 | DIMM turnaround |
| | | 1 | Read to write turnaround |
| | | 2 | Write to read turnaround |
| | | 7–3 | Reserved |
| E4h | NB | | Memory controller bypass counter saturation |
| | | 0 | Memory controller high priority bypass |
| | | 1 | Memory controller low priority bypass |
| | | 2 | DRAM controller interface bypass |
| | | 3 | DRAM controller queue bypass |
| | | 7–4 | Reserved |
| EBh | NB | | Sized commands |
| | | 0 | NonPostWrSzByte |
| | | 1 | NonPostWrSzDword |
| | | 2 | PostWrSzByte |

**Table 55. Performance Monitor Events (Continued)**

| EVENT_MASK [7:0] | Encoded EVENT_MASK [7:5] | UNIT_MASK bit | Description |
|---|---|---|---|
| | | 3 | PostWrSzDword |
| | | 4 | RdSzByte |
| | | 5 | RdSzDword |
| | | 6 | RdModWr |
| | | 7 | Reserved |
| ECh | NB | | Probe result |
| | | 0 | Probe miss |
| | | 1 | Probe hit |
| | | 2 | Probe hit dirty without memory cancel |
| | | 3 | Probe hit dirty with memory cancel |
| | | 7–4 | Reserved |
| F6h | NB | | HyperTransport™ bus 0 bandwidth |
| | | 0 | Command sent |
| | | 1 | Data sent |
| | | 2 | Buffer release sent |
| | | 3 | Nop sent |
| | | 7–4 | Reserved |
| F7h | NB | | HyperTransport™ bus 1 bandwidth |
| | | 0 | Command sent |
| | | 1 | Data sent |
| | | 2 | Buffer release sent |
| | | 3 | Nop sent |
| | | 7–4 | Reserved |
| F8h | NB | | HyperTransport™ bus 2 bandwidth |
| | | 0 | Command sent |
| | | 1 | Data sent |
| | | 2 | Buffer release sent |
| | | 3 | Nop sent |
| | | 7–4 | Reserved |

# 11 BIOS Checklist

The checklist in this chapter reviews information that is described elsewhere and is required by BIOS developers to properly incorporate AMD Athlon™ 64 and AMD Opteron™ processors into systems.

## 11.1 CPUID

Use the CPUID instruction to properly identify the AMD Athlon™ 64 processor or AMD Opteron™ processor.

- Determine the processor type, stepping, and available features by using functions 0000_0001h and 8000_0001h of the CPUID instruction.
- Boot-up display - The processor name should be displayed according to the processor name string defined in *CPUID Guide for the AMD Athlon™ 64 and AMD Opteron™ processors,* order# 25481.

For more information on the CPUID instruction, see *AMD Processor Recognition Application Note,* order# 20734 and *CPUID Guide for the AMD Athlon™ 64 and AMD Opteron™ processors,* order# 25481.

## 11.2 CPU Speed Detection

The BIOS can use the time-stamp counter (TSC) to clock a timed operation and compare the result to the Real-Time Clock (RTC) to determine the operating frequency. See the example of frequency-determination assembler code available on the AMD web site.

## 11.3 HyperTransport™ Link Frequency Selection

The BIOS should initialize HyperTransport™ link frequency (Link field, Function 0, Offsets 88h, A8h, C8h) with the minimum of the following frequencies:

- the maximum frequency defined by the link frequency capability bits (LnkFreqCap field, Function 0, Offsets 88h, A8h, C8h),
- the maximum frequency values specified in the processor and chipset data sheets,
- the maximum frequency supported by the platform.

## 11.4 Multiprocessing Capability Detection

The multiprocessing capability of the AMD Opteron™ processor is determined by the MPCap and BigMPCap bits in the Northbridge Capability Register (Function 3, Offset E8h).

| Multiprocessing Capability | MPCap | BigMPCap |
|---|---|---|
| UP Capable | 0 | 0 |
| DP Capable | 1 | 0 |
| MP Capable | 1 | 1 |

During POST, the BIOS checks the multiprocessing capability of AMD Opteron™ processors, and configures the system accordingly.

Multiprocessing capability detection is not required in a UP system.

All processors must be DP capable or MP capable in a DP system. If any processor is only UP capable, the BIOS must configure the BSP as a UP processor, and must not initialize the AP.

All processors must be MP capable in an MP system. If any processor is not MP capable, the BIOS must configure the BSP as a UP processor, and must not initialize APs.

If all processors do not have adequate multiprocessing capability for a DP or an MP system, the BIOS must display the following message:

```
********** Warning: non-MP Processor **********

The processor(s) installed in your system are not multiprocessing capable. Now
your system will halt.
```

If all processors have adequate multiprocessing capability for a DP or an MP system, but have different model numbers or operate at different frequencies, the BIOS must configure the BSP as a UP processor, and must not initialize APs. The BIOS must display the following message:

```
****** Warning: non-matching MP Processors ******

AMD multiprocessing-capable processors have been detected in your system.
However, all processors must be the same model and frequency to run in
multiprocessing mode. Now your system will halt.
```

After the error message is displayed, the BIOS must halt the system without further initialization.

## 11.5 Model-Specific Registers (MSRs)

Access only the MSRs that are implemented in the processor.

- Follow the processor MSR programming sequence described in this document. This includes setting the HWCR, SYSCFG, MTRRs, IORRs, clock control MSR, Top of Memory, and other registers.

## 11.6 Machine Check Architecture (MCA)

The processor supports a machine check architecture that allows reporting and handling of system errors through a consistent interface.

- The global MCA Status register must be cleared (all machine check features disabled).
- MC0_CTL must be set to all 1s because some do not set these bits.
- In addition, the local MCA status and address registers for the following five reporting blocks must also be cleared (all error and parity bits disabled):
  - BU—Bus Unit
  - IC—Instruction Cache Unit
  - DC—Data Cache Unit
  - LS—Load Store Unit
  - NB—Northbridge Unit

After reset, the BIOS needs to determine if the cause of the reset was a power-on or soft reset.

- When power-on reset occurs, the BIOS must clear the MCA registers.
- Otherwise, the BIOS should check for any MCA information that may be related to the soft reset.

For more information on MCA, see Chapter 5, "Machine Check Architecture."

## 11.7 Memory Map

### 11.7.1 I/O and Memory Type and Range Registers (IORRs, MTRRs)

The memory-type and range registers control the access and cacheability of memory regions in the processor.

- Follow the processor MTRR and IORR programming sequence that is described in this document.
- To initialize MTTRs in BIOS, set the default memory cache type for all addresses to uncacheable, then use a variable range MTRR to set all physical memory as cacheable (writeback), and finally use the fixed-range MTRRs to handle the special cases in the 640-Kbyte–1-Mbyte region. MTRRs should be modified while cache is disabled. Modifying MTRRs while cache is enabled will result in undefined behavior.
- The GART driver (miniport) should program the AGP aperture.
- The BIOS does not map the cacheability of the video frame buffer and AGP aperture space; it is done by the operating system, video, and AGP drivers.

### 11.7.2 Memory Map Registers (MMRs)

The processor contains several memory map registers to support the appropriate software view of memory.

- Program the correct values in the MSRs, such as Top of Memory (TOP_MEM).

## 11.8 Cache Testing and Programming

- BIOS must correctly maintain the tag parity and data ECC enable bits.

## 11.9 Memory System Configuration Registers

Node configuration space contains registers that define RAM, PCI memory, and x86 I/O address paths.

- Address mapping registers are contained in function 1 of each node.
- Cold boot and reset sends code fetch and other addresses to the compatibility link of Node0. At cold boot and reset, these mapping registers are disabled.
- Once enabled, these registers distribute addresses according to register mappings, which include node and link routing. See Chapter 8, "HyperTransport™ Technology Configuration and Enumeration."
- Thereafter, carefully set registers to reflect current needs for code fetch and memory access, i.e., do not map BIOS E0000h and F0000h space to DRAM until BIOS exits execution from the ROM chip. Otherwise, the system will hang. Likewise, memory-mapped I/O space should not be mapped above 1 Mbyte until BIOS exits execution from the ROM chip, or else the system will hang.
- Memory-mapped I/O (MMIO) and PCI I/O mappings must include address space needed to move option ROM code to RAM, as well as the required run-time resources. This could require setting and resetting the MMIO mappings during enumeration and device initialization. Configuration space access of a device with or without an option ROM is defined by the HyperTransport technology configuration mapping.
- Multihost bus multiprocessor systems require that MMIO mappings direct relevant addresses to the buses of devices through the node/link path to the bus.
- Application processors and the bootstrap processor in multiprocessor systems must be mapped separately with paths to target address space.

## 11.10 XSDT Table

The Extended System Description Table (XSDT) table defined in ACPI 2.0 must be implemented in the BIOS to support 64-bit operating systems for AMD Athlon™ 64 and AMD Opteron™ processor based systems.

## 11.11 Detect Target Operating Mode Callback

The operating system notifies the BIOS what the expected operating mode is with the Detect Target Operating Mode callback (INT 15, function EC00h). Based on the target operating mode, the BIOS can enable or disable mode specific performance and functional optimizations that are not visible to system software.

This callback does not change the operating mode; it only declares the target mode to the BIOS. It should be executed only once by the BSP before the first transition into long mode.

The default operating mode assumed by the BIOS is Legacy Mode Target Only. If this is not the target operating mode, system software must execute this callback to change it before transitioning to long mode for the first time. If the target operating mode is Legacy Mode Target Only, the callback does not need to be executed.

The Detect Target Operating Mode callback inputs are stored in the AX and BL registers. AX has a value of EC00h, selecting the Detect Target Operating Mode function. One of the following values in the BL register selects the operating mode:

- 01h — Legacy Mode Target Only. All enabled processors will operate in legacy mode only.
- 02h — Long Mode Target Only. All enabled processor will switch into long mode once.
- 03h — Mixed Mode Target. Processors may switch between legacy mode and long mode, or the preferred mode for system software is unknown. This value instructs the BIOS to use settings that are valid in all modes.
- All other values are reserved.

The Detect Target Operating Mode callback outputs are stored in the AH register and CF (carry flag in the EFLAGS register), and the values of other registers are not modified. The following output values are possible:

- AH = 00h and CF = 0, if the callback is implemented and the value in BL is supported.
- AH = 00h and CF = 1, if the callback is implemented and the value in BL is reserved. This indicates an error; the target operating mode is set to Legacy Mode Target Only.
- AH = 86h and CF = 1, if the callback is not supported.

## 11.12 SMM Issues

The processor includes support for system management mode (SMM). The functionality of the AMD Athlon™ 64 processor or AMD Opteron™ processor SMM is a superset of the Pentium® processor functionality.

- The AMD Athlon™ 64 and AMD Opteron™ processors implement the SMM remapping and control registers as model-specific registers on the CPU.
- Implement the SMM state-save area in a manner compatible with the description of SMM found in Chapter 6, "System Management Mode (SMM)."

Program the model-specific registers to set the SMM memory base, local address, destination address, memory type, size and control, etc. See Chapter 6, "System Management Mode (SMM)."

# 12 Processor Configuration Registers

This chapter includes descriptions of two types of model-specific registers, as follows:

- General model-specific registers (see page 261)
- AMD Athlon™ 64 and AMD Opteron™ model-specific registers (see page 284)

## 12.1 General Model-Specific Registers

Table 56 is a listing of the general model-specific registers supported by the processor, presented in ascending hexadecimal address order. Register descriptions follow the table, organized according to the following functions:

- System software (see page 263)
- Memory typing (see page 265)
- APIC (see page 282)
- Machine check architecture (see page 146)
- Software debug (see page 283)
- Performance monitoring (see page 284)

**Table 56. General MSRs**

| Address | Register Name | Description |
|---|---|---|
| 0010h | TSC | page 284 |
| 001Bh | APIC_BASE | page 282 |
| 002Ah | EBL_CR_POWERON | page 282 |
| 00FEh | MTRRcap | page 265 |
| 0174h | SYSENTER_CS | page 263 |
| 0175h | SYSENTER_ESP | page 264 |
| 0176h | SYSENTER_EIP | page 264 |
| 0179h | MCG_CAP | page 147 |
| 017Ah | MCG_STATUS | page 147 |
| 017Bh | MCG_CTL | page 148 |

**Table 56. General MSRs (Continued)**

| Address | Register Name | Description |
|---|---|---|
| 01D9h | DebugCtl | page 283 |
| 01DBh | LastBranchFromIP | page 283 |
| 01DCh | LastBranchToIP | page 283 |
| 01DDh | LastExceptionFromIP | page 283 |
| 01DEh | LastExceptionToIP | page 283 |
| 0200–020Eh Even | MTRRphysBase[7:0] | page 266 |
| 0201–020Fh Odd | MTRRphysMask[7:0] | page 266 |
| 0250h | MTRRfix64K_00000 | page 267 |
| 0258h | MTRRfix16K_80000 | page 268 |
| 0259h | MTRRfix16K_A0000 | page 269 |
| 0268h | MTRRfix4K_C0000 | page 271 |
| 0269h | MTRRfix4K_C8000 | page 272 |
| 026Ah | MTRRfix4K_D0000 | page 273 |
| 026Bh | MTRRfix4K_D8000 | page 274 |
| 026Ch | MTRRfix4K_E0000 | page 275 |
| 026Dh | MTRRfix4K_E8000 | page 276 |
| 026Eh | MTRRfix4K_F0000 | page 278 |
| 026Fh | MTRRfix4K_F8000 | page 279 |
| 0277h | PAT | page 280 |
| 02FFh | MTRRdefType | page 281 |
| 0400h | MC0_CTL | page 152 |
| 0401h | MC0_STATUS | page 153 |
| 0402h | MC0_ADDR | page 155 |
| 0403h, | MC0_MISC | not supported |
| 0404h | MC1_CTL | page 156 |
| 0405h | MC1_STATUS | page 157 |
| 0406h | MC1_ADDR | page 157 |
| 0407h | MC1_MISC | not supported |

**Table 56. General MSRs (Continued)**

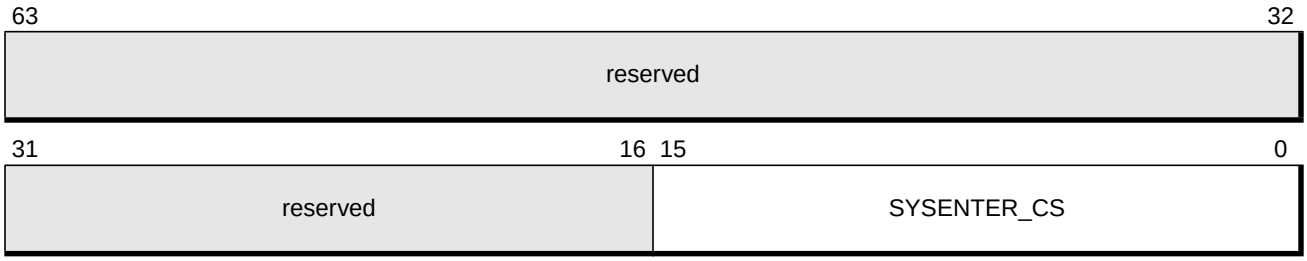
| Address | Register Name | Description |
|---|---|---|
| 0408h | MC2_CTL | page 156 |
| 0409h | MC2_STATUS | page 160 |
| 040Ah | MC2_ADDR | page 160 |
| 040Bh | MC2_MISC | not supported |
| 040Ch | MC3_CTL | page 161 |
| 040Dh | MC3_STATUS | page 162 |
| 040Eh | MC3_ADDR | page 162 |
| 040Fh | MC3_MISC | not supported |
| 0410h | MC4_CTL | page 162 |
| 0411h | MC4_STATUS | page 162 |
| 0412h | MC4_ADDR | page 162 |
| 0413h | MC4_MISC | not supported |

## 12.1.1 System Software Registers

### 12.1.1.1 SYSENTER_CS Register

This register contains the code segment selector used by the SYSENTER and SYSEXIT instructions. See the SYSENTER and SYSEXIT section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

**SYSENTER_CS Register** **MSR 0174h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–32 | reserved | RAZ | R | 0 |
| 31–16 | reserved | SBZ | R/W | 0 |
| 15–0 | SYSENTER_CS | SYSENTER/SYSEXIT code segment selector | R/W | 0 |

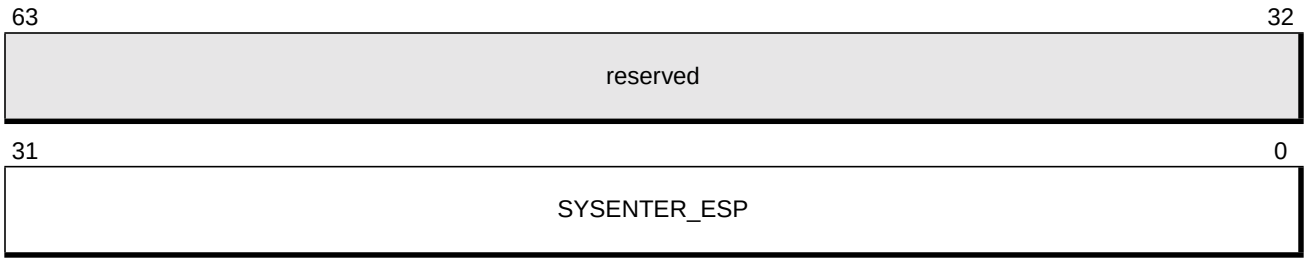
### Field Descriptions

**SYSENTER/SYSEXIT Code Segment Selector (SYSENTER_CS)**—Bits 15–0.

### 12.1.1.2 SYSENTER_ESP Register

This register contains the stack pointer used by the SYSENTER and SYSEXIT instructions. See the SYSENTER and SYSEXIT section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

**SYSENTER_ESP Register** **MSR 0175h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–32 | reserved | RAZ | R | 0 |
| 31–0 | SYSENTER_ESP | SYSENTER/SYSEXIT stack pointer | R/W | 0 |

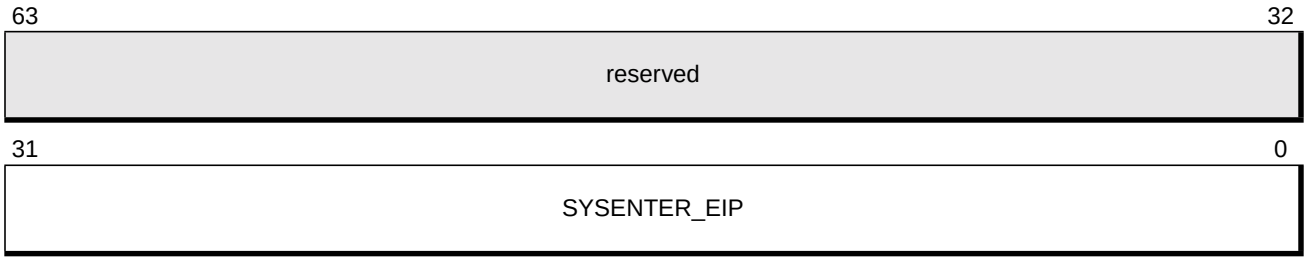
### Field Descriptions

**SYSENTER/SYSEXIT Stack Pointer (SYSENTER_ESP)**—Bits 31–0.

### 12.1.1.3 SYSENTER_EIP Register

This register contains the instruction pointer used by the SYSENTER and SYSEXIT instructions. See the SYSENTER and SYSEXIT section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

**SYSENTER_EIP Register** **MSR 0176h**



| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–32 | reserved | RAZ | R | 0 |
| 31–0 | SYSENTER_EIP | SYSENTER/SYSEXIT instruction pointer | R/W | 0 |

### Field Descriptions

**SYSENTER/SYSEXIT Instruction Pointer (SYSENTER_EIP)**—Bits 31–0.

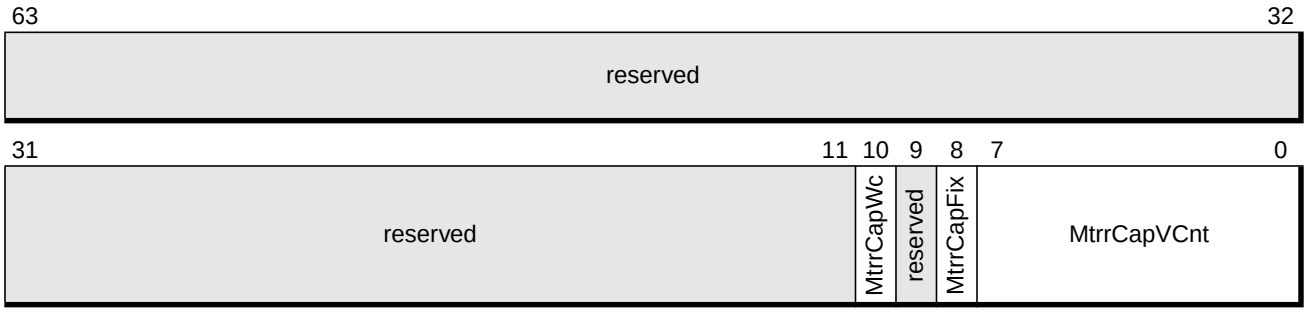
## 12.1.2 Memory Typing Registers

### 12.1.2.1 MTRRcap Register

This is a read-only register that returns information about the processors MTRR capabilities. See the "Using MTRRs" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

The MTRRcap register is a read-only status register. Attempting to modify this register will result in a #GP(0).

**MTRRcap Register** **MSR 00FEh**




| Bit | Mnemonic | Function | R/W |
|---|---|---|---|
| 63–11 | reserved | RAZ | R |
| 10 | MtrrCapWc | Write-combining memory type | R |
| 9 | reserved | RAZ | R |
| 8 | MtrrCapFix | Fixed range register | R |
| 7–0 | MtrrCapVCnt | Variable range registers count | R |

### Field Descriptions

**Variable Range Registers Count (MtrrCapVCnt)**—Bits 7–0. Indicates number of variable range registers.

**Fixed Range Registers (MtrrCapFix)**—Bit 8. Indicates fixed range register capability.

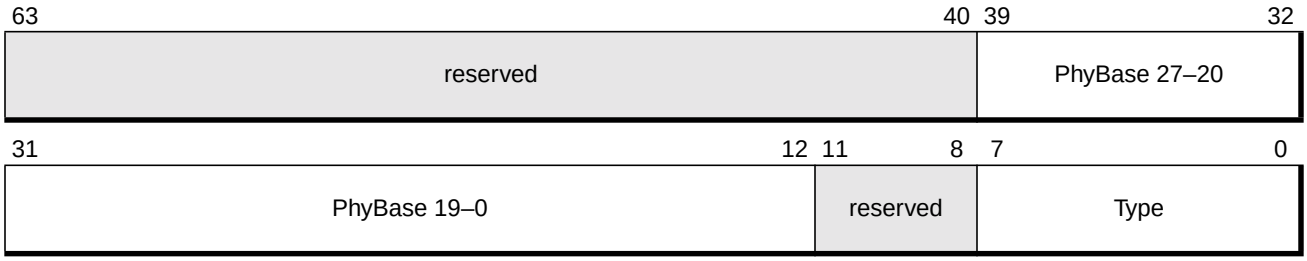
**Write-Combining Memory Type (MtrrCapWc)**—Bit 10. Indicates write-combining memory type capability.

### 12.1.2.2 MTRRphysBase*i* Registers

These registers define the base address and memory type for each of the variable MTRRs. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Attempting to modify reserved bits in MTRRphysBase*i* will result in a #GP(0).

**MTRRphysBase0–7 Registers** — **MSRs 0200h, 0202h, 0204h, 0206h, 0208h, 020Ah, 020Ch, 020Eh**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–40 | reserved | MBZ | | 0 |
| 39–12 | PhyBase | Base address | R/W | U |
| 11–8 | reserved | MBZ | | 0 |
| 7–0 | Type | Memory type | R/W | U |

#### Field Descriptions

**Memory Type (Type)**—Bits 7–0. Specifies memory type for this memory range.

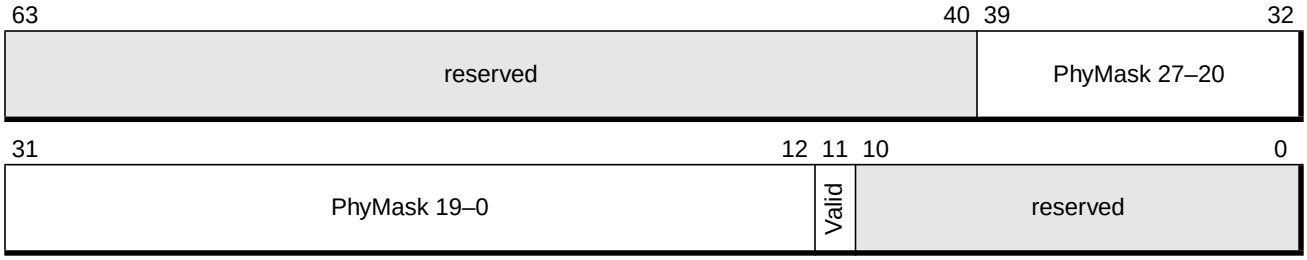
**Base Address (PhysBase)**—Bits 39–12. Specifies base address for this memory range

### 12.1.2.3 MTRRphysMask*i* Registers

These registers define the address mask and valid bit for each of the variable MTRRs. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Attempting to modify reserved bits in MTRRphysMask*i* will result in a #GP(0).

### MTRRphysMask0–7 Registers — MSRs 0201h, 0203h, 0205, 0207h, 0209h, 020Bh, 020Dh, 020Fh




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–40 | reserved | MBZ | | 0 |
| 39–12 | PhysMask | Address mask | R/W | U |
| 11 | Valid | MTRR is valid | R/W | 0 |
| 10–0 | reserved | MBZ | | 0 |

## Field Descriptions

**MTRR is Valid (Valid)**—Bit 11. Indicated MTRR is valid.

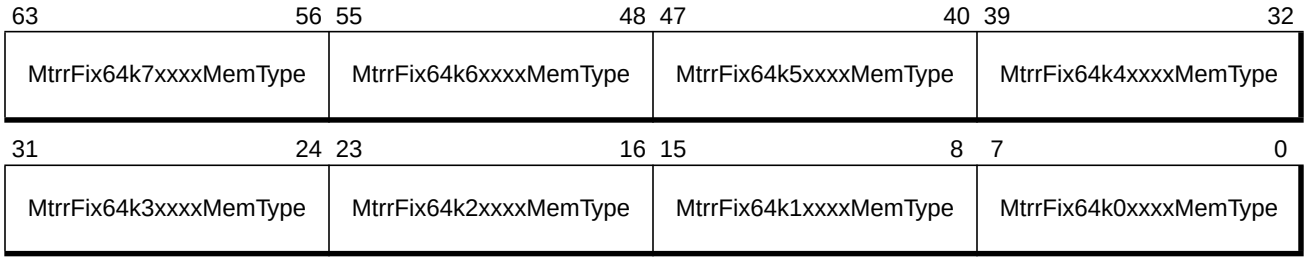
**Address Mask (PhysMask)**—Bits 39–12. Specifies the address mask for this memory range.

### 12.1.2.4 MTRRfix64K_00000 Register

This register controls the memory types for the first 512 Kbyte of physical memory. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

### MTRRfix64K_00000 Register — MSR 0250h




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix64k7xxxxMemType | Memory type address 70000–7FFFFh | R/W | U |
| 55–48 | MtrrFix64k6xxxxMemType | Memory type address 60000–6FFFFh | R/W | U |
| 47–40 | MtrrFix64k5xxxxMemType | Memory type address 50000–5FFFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 39–32 | MtrrFix64k4xxxxMemType | Memory type address 40000–4FFFFh | R/W | U |
| 31–24 | MtrrFix64k3xxxxMemType | Memory type address 30000–3FFFFh | R/W | U |
| 23–16 | MtrrFix64k2xxxxMemType | Memory type address 20000–2FFFFh | R/W | U |
| 15–8 | MtrrFix64k1xxxxMemType | Memory type address 10000–1FFFFh | R/W | U |
| 7–0 | MtrrFix64k0xxxxMemType | Memory type address 00000–0FFFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Memory Type Address 00000–0FFFFh (MtrrFix64k0xxxxMemType)**—Bits 7–0. Memory type for physical address 00000–0FFFFh.

**Memory Type Address 10000–1FFFFh (MtrrFix64k1xxxxMemType)**—Bits 15–8. Memory type for physical address 10000–1FFFFh.

**Memory Type Address 20000–2FFFFh (MtrrFix64k2xxxxMemType)**—Bits 23–16. Memory type for physical address 20000–2FFFFh.

**Memory Type Address 30000–3FFFFh (MtrrFix64k3xxxxMemType)**—Bits 31–24. Memory type for physical address 30000–3FFFFh.

**Memory Type Address 40000–4FFFFh (MtrrFix64k4xxxxMemType)**—Bits 39–32. Memory type for physical address 40000–4FFFFh.

**Memory Type Address 50000–5FFFFh (MtrrFix64k5xxxxMemType)**—Bits 47–40. Memory type for physical address 50000–5FFFFh.

**Memory Type Address 60000–6FFFFh (MtrrFix64k6xxxxMemType)**—Bits 55–48. Memory type for physical address 60000–6FFFFh.

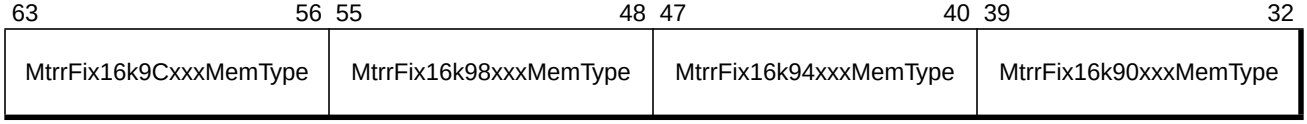
**Memory Type Address 70000–7FFFFh (MtrrFix64k7xxxxMemType)**—Bits 63–56. Memory type for physical address 70000–7FFFFh.

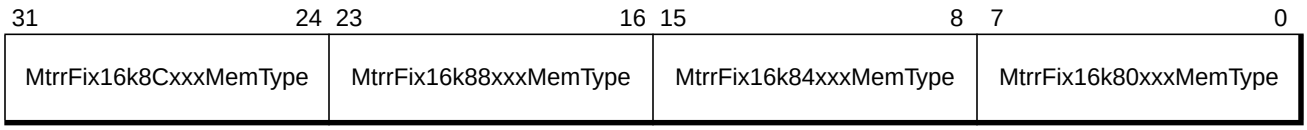
#### 12.1.2.5 MTRRfix16K_80000 Register

This register controls the memory types for physical memory addresses 80000–9FFFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix16K_80000 Register** **MSR 0258h**






| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix16k9CxxxMemType | Memory type address 9C000–9FFFFh | R/W | U |
| 55–48 | MtrrFix16k98xxxMemType | Memory type address 98000–9BFFFh | R/W | U |
| 47–40 | MtrrFix16k94xxxMemType | Memory type address 94000–97FFFh | R/W | U |
| 39–32 | MtrrFix16k90xxxMemType | Memory type address 90000–93FFFh | R/W | U |
| 31–24 | MtrrFix16k8CxxxMemType | Memory type address 8C000–8FFFFh | R/W | U |
| 23–16 | MtrrFix16k88xxxMemType | Memory type address 88000–8BFFFh | R/W | U |
| 15–8 | MtrrFix16k84xxxMemType | Memory type address 84000–87FFFh | R/W | U |
| 7–0 | MtrrFix16k80xxxMemType | Memory type address 80000–83FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

## Field Descriptions

**Memory Type Address 80000–83FFFh (MtrrFix16k80xxxMemType)**—Bits 7–0. Memory type for physical address 80000–83FFFh.

**Memory Type Address 84000–87FFFh (MtrrFix16k84xxxMemType)**—Bits 15–8. Memory type for physical address 84000–87FFFh.

**Memory Type Address 88000–8BFFFh (MtrrFix16k88xxxMemType)**—Bits 23–16. Memory type for physical address 88000–8BFFFh.

**Memory Type Address 8C000–8FFFFh (MtrrFix16k8CxxxMemType)**—Bits 31–24. Memory type for physical address 8C000–8FFFFh.

**Memory Type Address 90000–93FFFh (MtrrFix16k90xxxMemType)**—Bits 39–32. Memory type for physical address 90000–93FFFh.

**Memory Type Address 94000–97FFFh (MtrrFix16k94xxxMemType)**—Bits 47–40. Memory type for physical address 94000–97FFFh.

**Memory Type Address 98000–9BFFFh (MtrrFix16k98xxxMemType)**—Bits 55–48. Memory type for physical address 98000–9BFFFh.

**Memory Type Address 9C000–9FFFFh (MtrrFix16k9CxxxMemType)**—Bits 63–56. Memory type for physical address 9C000–9FFFFh.

### 12.1.2.6 MTRRfix16K_A0000 Register

This register controls the memory types for physical memory addresses A0000–BFFFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

## MTRRfix16K_A0000 Register — MSR 0259h

| 63 – 56 | 55 – 48 | 47 – 40 | 39 – 32 |
|---|---|---|---|
| MtrrFix16kBCxxxMemType | MtrrFix16kB8xxxMemType | MtrrFix16kB4xxxMemType | MtrrFix16kB0xxxMemType |
| **31 – 24** | **23 – 16** | **15 – 8** | **7 – 0** |
| MtrrFix16kACxxxMemType | MtrrFix16kA8xxxMemType | MtrrFix16kA4xxxMemType | MtrrFix16kA0xxxMemType |

| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix16kBCxxxMemType | Memory type address BC000–BFFFFh | R/W | U |
| 55–48 | MtrrFix16kB8xxxMemType | Memory type address B8000–BBFFFh | R/W | U |
| 47–40 | MtrrFix16kB4xxxMemType | Memory type address B4000–B7FFFh | R/W | U |
| 39–32 | MtrrFix16kB0xxxMemType | Memory type address B0000–B3FFFh | R/W | U |
| 31–24 | MtrrFix16kACxxxMemType | Memory type address AC000–AFFFFh | R/W | U |
| 23–16 | MtrrFix16kA8xxxMemType | Memory type address A8000–ABFFFh | R/W | U |
| 15–8 | MtrrFix16kA4xxxMemType | Memory type address A4000–A7FFFh | R/W | U |
| 7–0 | MtrrFix16kA0xxxMemType | Memory type address A0000–A3FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Memory Type Address A0000–A3FFFh (MtrrFix16kA0xxxMemType)**—Bits 7–0. Memory type for physical address A0000–A3FFFh.

**Memory Type Address A4000–A7FFFh (MtrrFix16kA4xxxMemType)**—Bits 15–8. Memory type for physical address A4000–A7FFFh.

**Memory Type Address A8000–ABFFFh (MtrrFix16kA8xxxMemType)**—Bits 23–16. Memory type for physical address A8000–ABFFFh.

**Memory Type Address AC000–AFFFFh (MtrrFix16kACxxxMemType)**—Bits 31–24. Memory type for physical address AC000–AFFFFh.

**Memory Type Address B0000–B3FFFh (MtrrFix16kB0xxxMemType)**—Bits 39–32. Memory type for physical address B0000–B3FFFh.

**Memory Type Address B4000–B7FFFh (MtrrFix16kB4xxxMemType)**—Bits 47–40. Memory type for physical address B4000–B7FFFh.

**Memory Type Address B8000–BBFFFh (MtrrFix16kB8xxxMemType)**—Bits 55–48. Memory type for physical address B8000–BBFFFh.

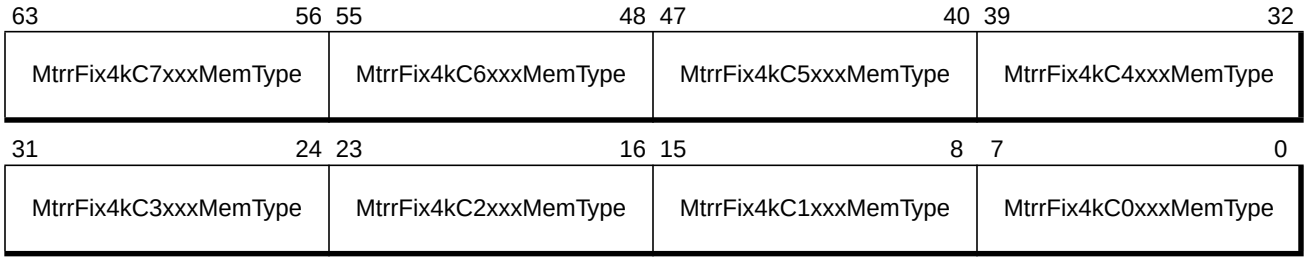
**Memory Type Address BC000–BFFFFh (MtrrFix16kBCxxxMemType)**—Bits 63–56. Memory type for physical address BC000–BFFFFh.

### 12.1.2.7 MTRRfix4K_C0000 Register

This register controls the memory types for physical memory addresses C0000–C7FFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K_C0000 Register** **MSR 0268h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kC7xxxMemType | Memory type address C7000–C7FFFh | R/W | U |
| 55–48 | MtrrFix4kC6xxxMemType | Memory type address C6000–C6FFFh | R/W | U |
| 47–40 | MtrrFix4kC5xxxMemType | Memory type address C5000–C5FFFh | R/W | U |
| 39–32 | MtrrFix4kC4xxxMemType | Memory type address C4000–C4FFFh | R/W | U |
| 31–24 | MtrrFix4kC3xxxMemType | Memory type address C3000–C3FFFh | R/W | U |
| 23–16 | MtrrFix4kC2xxxMemType | Memory type address C2000–C2FFFh | R/W | U |
| 15–8 | MtrrFix4kC1xxxMemType | Memory type address C1000–C1FFFh | R/W | U |
| 7–0 | MtrrFix4kC0xxxMemType | Memory type address C0000–C0FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

#### Field Descriptions

**Memory Type Address C0000–C0FFFh (MtrrFix4kC0xxxMemType)**—Bits 7–0. Memory type for physical address C0000–C0FFFh.

**Memory Type Address C1000–C1FFFh (MtrrFix4kC1xxxMemType)**—Bits 15–8. Memory type for physical address C1000–C1FFFh.

**Memory Type Address C2000–C2FFFh (MtrrFix4kC2xxxMemType)**—Bits 23–16. Memory type for physical address C2000–C2FFFh.

**Memory Type Address C3000–C3FFFh (MtrrFix4kC3xxxMemType)**—Bits 31–24. Memory type for physical address C3000–C3FFFh.

**Memory Type Address C40000–C4FFFh (MtrrFix4kC4xxxMemType)**—Bits 39–32. Memory type for physical address C4000–C4FFFh.

**Memory Type Address C5000–C5FFFh (MtrrFix4kC5xxxMemType)**—Bits 47–40. Memory type for physical address C5000–C5FFFh.

**Memory Type Address C6000–C6FFFh (MtrrFix4kC6xxxMemType)**—Bits 55–48. Memory type for physical address C6000–C6FFFh.

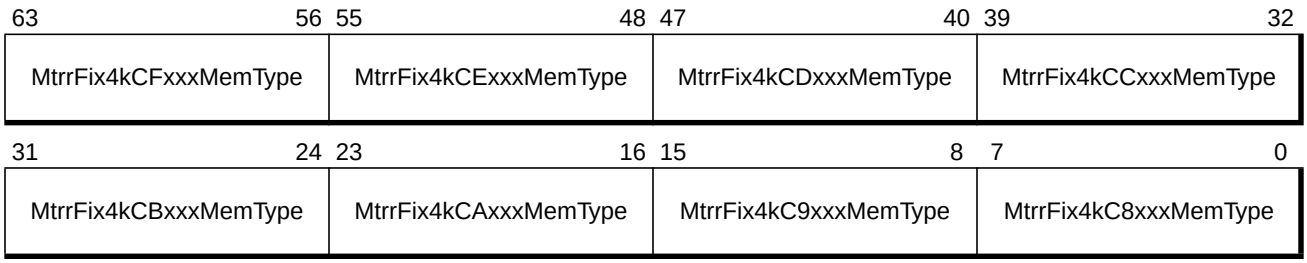
**Memory Type Address C7000–C7FFFh (MtrrFix4kC7xxxMemType)**—Bits 63–56. Memory type for physical address C7000–C7FFFh.

### 12.1.2.8 MTRRfix4K_C8000 Register

This register controls the memory types for physical memory addresses C8000–CFFFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K_C8000 Register** **MSR 0269h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kCFxxxMemType | Memory type for address CF000–CFFFFh | R/W | U |
| 55–48 | MtrrFix4kCExxxMemType | Memory type for address CE000–CEFFFh | R/W | U |
| 47–40 | MtrrFix4kCDxxxMemType | Memory type for address CD000–CDFFFh | R/W | U |
| 39–32 | MtrrFix4kCCxxxMemType | Memory type for address CC000–CCFFFh | R/W | U |
| 31–24 | MtrrFix4kCBxxxMemType | Memory type for address CB000–CBFFFh | R/W | U |
| 23–16 | MtrrFix4kCAxxxMemType | Memory type for address CA000–CAFFFh | R/W | U |
| 15–8 | MtrrFix4kC9xxxMemType | Memory type for address C9000–C9FFFh | R/W | U |
| 7–0 | MtrrFix4kC8xxxMemType | Memory type for address C8000–C8FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

#### Field Descriptions

**Memory Type Address C8000–C8FFFh (MtrrFix4kC8xxxMemType)**—Bits 7–0. Memory type for physical address C8000–C8FFFh.

**Memory Type Address C9000–C9FFFh (MtrrFix4kC9xxxMemType)**—Bits 15–8. Memory type for physical address C9000–C9FFFh.

**Memory Type Address CA000–CAFFFh (MtrrFix4kCAxxxMemType)**—Bits 23–16. Memory type for physical address CA000–CAFFFh.

**Memory Type Address CB000–CBFFFh (MtrrFix4kCBxxxMemType)**—Bits 31–24. Memory type for physical address CB000–CBFFFh.

**Memory Type Address CC0000–CCFFFh (MtrrFix4kCCxxxMemType)**—Bits 39–32. Memory type for physical address CC000–CCFFFh.

**Memory Type Address CD000–CDFFFh (MtrrFix4kCDxxxMemType)**—Bits 47–40. Memory type for physical address CD000–CDFFFh.

**Memory Type Address CE000–CEFFFh (MtrrFix4kCExxxMemType)**—Bits 55–48. Memory type for physical address CE000–CEFFFh.

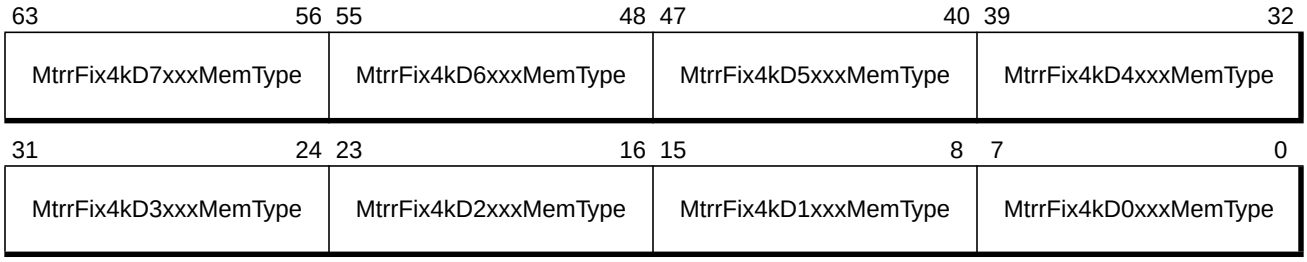
**Memory Type Address CF000–CFFFFh (MtrrFix4kCFxxxMemType)**—Bits 63–56. Memory type for physical address CF000–CFFFFh.

### 12.1.2.9 MTRRfix4K_D0000 Register

This register controls the memory types for physical memory addresses D0000–D7FFF. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K_D0000 Register** **MSR 026Ah**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kD7xxxMemType | Memory type address D7000–D7FFFh | R/W | U |
| 55–48 | MtrrFix4kD6xxxMemType | Memory type address D6000–D6FFFh | R/W | U |
| 47–40 | MtrrFix4kD5xxxMemType | Memory type address D5000–D5FFFh | R/W | U |
| 39–32 | MtrrFix4kD4xxxMemType | Memory type address D4000–D4FFFh | R/W | U |
| 31–24 | MtrrFix4kD3xxxMemType | Memory type address D3000–D3FFFh | R/W | U |
| 23–16 | MtrrFix4kD2xxxMemType | Memory type address D2000–D2FFFh | R/W | U |
| 15–8 | MtrrFix4kD1xxxMemType | Memory type address D1000–D1FFFh | R/W | U |
| 7–0 | MtrrFix4kD0xxxMemType | Memory type address D0000–D0FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Memory Type Address D0000–D0FFFh (MtrrFix4kD0xxxMemType)**—Bits 7–0. Memory type for physical address D0000–D0FFFh.

**Memory Type Address D1000–D1FFFh (MtrrFix4kD1xxxMemType)**—Bits 15–8. Memory type for physical address D1000–D1FFFh.

**Memory Type Address D2000–D2FFFh (MtrrFix4kD2xxxMemType)**—Bits 23–16. Memory type for physical address D2000–D2FFFh.

**Memory Type Address D3000–D3FFFh (MtrrFix4kD3xxxMemType)**—Bits 31–24. Memory type for physical address D3000–D3FFFh.

**Memory Type Address D40000–D4FFFh (MtrrFix4kD4xxxMemType)**—Bits 39–32. Memory type for physical address D4000–D4FFFh.

**Memory Type Address D5000–D5FFFh (MtrrFix4kD5xxxMemType)**—Bits 47–40. Memory type for physical address D5000–D5FFFh.

**Memory Type Address D6000–D6FFFh (MtrrFix4kD6xxxMemType)**—Bits 55–48. Memory type for physical address D6000–D6FFFh.

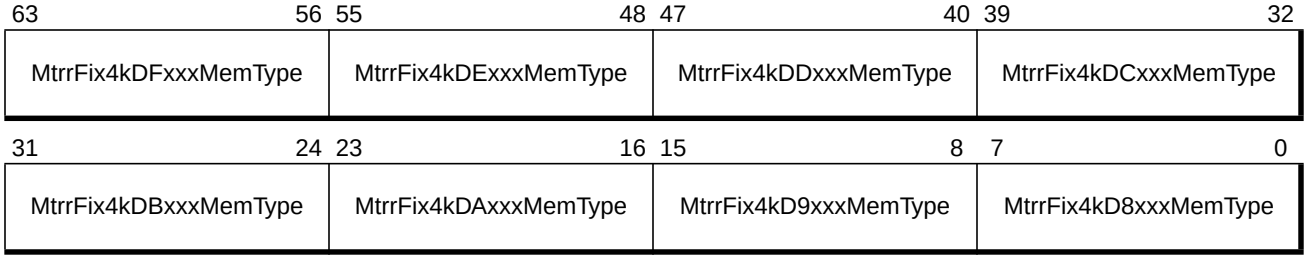
**Memory Type Address D7000–D7FFFh (MtrrFix4kD7xxxMemType)**—Bits 63–56. Memory type for physical address D7000–D7FFFh.

### 12.1.2.10 MTRRfix4K_D8000 Register

This register controls the memory types for physical memory addresses D8000–DFFFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K_D8000 Register** **MSR 026Bh**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kDFxxxMemType | Memory type address DF000–DFFFFh | R/W | U |
| 55–48 | MtrrFix4kDExxxMemType | Memory type address DE000–DEFFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 47–40 | MtrrFix4kDDxxxMemType | Memory type address DD000–DDFFFh | R/W | U |
| 39–32 | MtrrFix4kDCxxxMemType | Memory type address DC000–DCFFFh | R/W | U |
| 31–24 | MtrrFix4kDBxxxMemType | Memory type address DB000–DBFFFh | R/W | U |
| 23–16 | MtrrFix4kDAxxxMemType | Memory type address DA000–DAFFFh | R/W | U |
| 15–8 | MtrrFix4kD9xxxMemType | Memory type address D9000–D9FFFh | R/W | U |
| 7–0 | MtrrFix4kD8xxxMemType | Memory type address D8000–D8FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Memory Type Address D8000–D8FFFh (MtrrFix4kD8xxxMemType)**—Bits 7–0. Memory type for physical address D8000–D8FFFh.

**Memory Type Address D9000–D9FFFh (MtrrFix4kD9xxxMemType)**—Bits 15–8. Memory type for physical address D9000–D9FFFh.

**Memory Type Address DA000–DAFFFh (MtrrFix4kDAxxxMemType)**—Bits 23–16. Memory type for physical address DA000–DAFFFh.

**Memory Type Address DB000–DBFFFh (MtrrFix4kDBxxxMemType)**—Bits 31–24. Memory type for physical address DB000–DBFFFh.

**Memory Type Address DC0000–DCFFFh (MtrrFix4kDCxxxMemType)**—Bits 39–32. Memory type for physical address DC000–DCFFFh.

**Memory Type Address DD000–DDFFFh (MtrrFix4kDDxxxMemType)**—Bits 47–40. Memory type for physical address DD000–DDFFFh.

**Memory Type Address DE000–DEFFFh (MtrrFix4kDExxxMemType)**—Bits 55–48. Memory type for physical address DE000–DEFFFh.

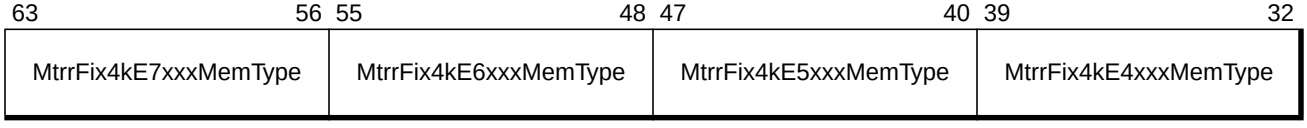
**Memory Type Address DF000–DFFFFh (MtrrFix4kDFxxxMemType)**—Bits 63–56. Memory type for physical address DF000–DFFFFh.

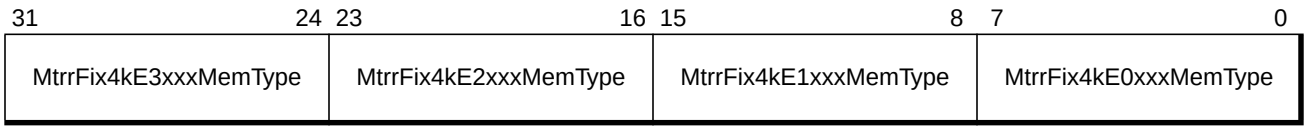
### 12.1.2.11 MTRRfix4K_E0000 Register

This register controls the memory types for physical memory addresses E0000–E7FFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K_E0000 Register** **MSR 026Ch**






| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kE7xxxMemType | Memory type for physical addr E7000–E7FFFh | R/W | U |
| 55–48 | MtrrFix4kE6xxxMemType | Memory type for physical addr E6000–E6FFFh | R/W | U |
| 47–40 | MtrrFix4kE5xxxMemType | Memory type for physical addr E5000–E5FFFh | R/W | U |
| 39–32 | MtrrFix4kE4xxxMemType | Memory type for physical addr E4000–E4FFFh | R/W | U |
| 31–24 | MtrrFix4kE3xxxMemType | Memory type for physical addr E3000–E3FFFh | R/W | U |
| 23–16 | MtrrFix4kE2xxxMemType | Memory type for physical addr E2000–E2FFFh | R/W | U |
| 15–8 | MtrrFix4kE1xxxMemType | Memory type for physical addr E1000–E1FFFh | R/W | U |
| 7–0 | MtrrFix4kE0xxxMemType | Memory type for physical addr E0000–E0FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

## Field Descriptions

**Memory Type Address E0000–E0FFFh (MtrrFix4kE0xxxMemType)**—Bits 7–0. Memory type for physical address E0000–E0FFFh.

**Memory Type Address E1000–E1FFFh (MtrrFix4kE1xxxMemType)**—Bits 15–8. Memory type for physical address E1000–E1FFFh.

**Memory Type Address E2000–E2FFFh (MtrrFix4kE2xxxMemType)**—Bits 23–16. Memory type for physical address E2000–E2FFFh.

**Memory Type Address E3000–E3FFFh (MtrrFix4kE3xxxMemType)**—Bits 31–24. Memory type for physical address E3000–E3FFFh.

**Memory Type Address E40000–E4FFFh (MtrrFix4kE4xxxMemType)**—Bits 39–32. Memory type for physical address E4000–E4FFFh.

**Memory Type Address E5000–E5FFFh (MtrrFix4kE5xxxMemType)**—Bits 47–40. Memory type for physical address E5000–E5FFFh.

**Memory Type Address E6000–E6FFFh (MtrrFix4kE6xxxMemType)**—Bits 55–48. Memory type for physical address E6000–E6FFFh.

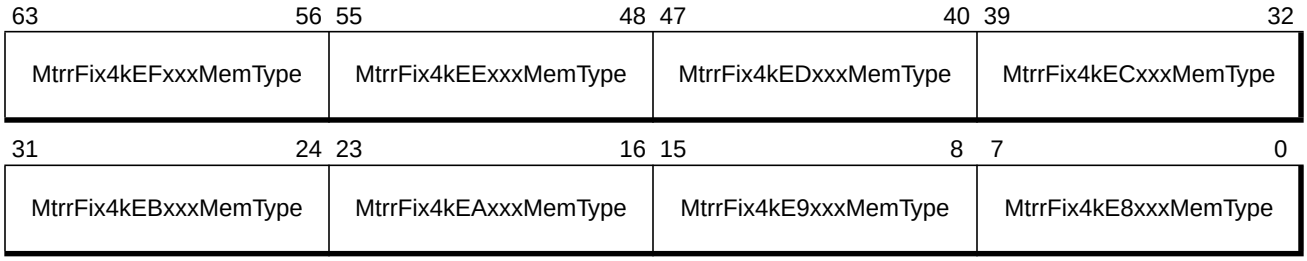
**Memory Type Address E7000–E7FFFh (MtrrFix4kE7xxxMemType)**—Bits 63–56. Memory type for physical address E7000–E7FFFh.

### 12.1.2.12 MTRRfix4K_E8000 Register

This register controls the memory types for physical memory addresses E8000–EFFFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

## MTRRfix4K_E8000 Register — MSR 026Dh




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kEFxxxMemType | Memory type for physical addr EF000–EFFFFh | R/W | U |
| 55–48 | MtrrFix4kEExxxMemType | Memory type for physical addr EE000–EEFFFh | R/W | U |
| 47–40 | MtrrFix4kEDxxxMemType | Memory type for physical addr ED000–EDFFFh | R/W | U |
| 39–32 | MtrrFix4kECxxxMemType | Memory type for physical addr EC000–ECFFFh | R/W | U |
| 31–24 | MtrrFix4kEBxxxMemType | Memory type for physical addr EB000–EBFFFh | R/W | U |
| 23–16 | MtrrFix4kEAxxxMemType | Memory type for physical addr EA000–EAFFFh | R/W | U |
| 15–8 | MtrrFix4kE9xxxMemType | Memory type for physical addr E9000–E9FFFh | R/W | U |
| 7–0 | MtrrFix4kE8xxxMemType | Memory type for physical addr E8000–E8FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Memory Type Address E8000–E8FFFh (MtrrFix4kE8xxxMemType)**—Bits 7–0. Memory type for physical address E8000–E8FFFh.

**Memory Type Address E9000–E9FFFh (MtrrFix4kE9xxxMemType)**—Bits 15–8. Memory type for physical address E9000–E9FFFh.

**Memory Type Address EA000–EAFFFh (MtrrFix4kEAxxxMemType)**—Bits 23–16. Memory type for physical address EA000–EAFFFh.

**Memory Type Address EB000–EBFFFh (MtrrFix4kEBxxxMemType)**—Bits 31–24. Memory type for physical address EB000–EBFFFh.

**Memory Type Address EC0000–ECFFFh (MtrrFix4kECxxxMemType)**—Bits 39–32. Memory type for physical address EC000–ECFFFh.

**Memory Type Address ED000–EDFFFh (MtrrFix4kEDxxxMemType)**—Bits 47–40. Memory type for physical address ED000–EDFFFh.

**Memory Type Address EE000–EEFFFh (MtrrFix4kEExxxMemType)**—Bits 55–48. Memory type for physical address EE000–EEFFFh.

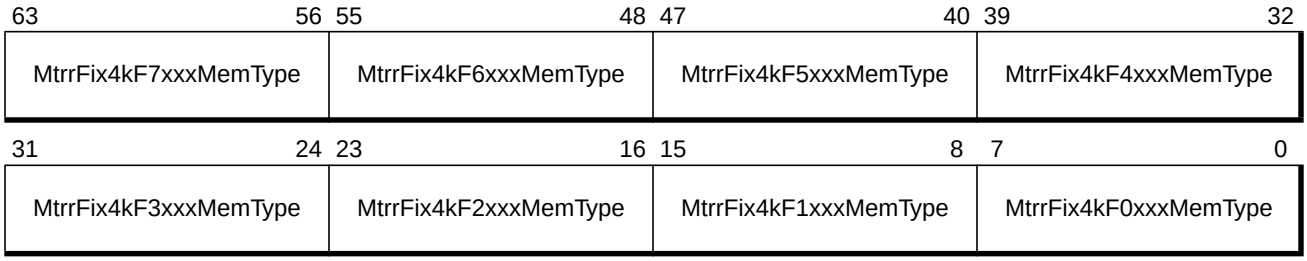
**Memory Type Address EF000–EFFFFh (MtrrFix4kEFxxxMemType)**—Bits 63–56. Memory type for physical address EF000–EFFFFh.

### 12.1.2.13 MTRRfix4K_F0000 Register

This register controls the memory types for physical memory addresses F0000–F7FFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K_F0000 Register** **MSR 026Eh**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kF7xxxMemType | Memory type address F7000–F7FFFh | R/W | U |
| 55–48 | MtrrFix4kF6xxxMemType | Memory type address F6000–F6FFFh | R/W | U |
| 47–40 | MtrrFix4kF5xxxMemType | Memory type address F5000–F5FFFh | R/W | U |
| 39–32 | MtrrFix4kF4xxxMemType | Memory type address F4000–F4FFFh | R/W | U |
| 31–24 | MtrrFix4kF3xxxMemType | Memory type address F3000–F3FFFh | R/W | U |
| 23–16 | MtrrFix4kF2xxxMemType | Memory type address F2000–F2FFFh | R/W | U |
| 15–8 | MtrrFix4kF1xxxMemType | Memory type address F1000–F1FFFh | R/W | U |
| 7–0 | MtrrFix4kF0xxxMemType | Memory type address F0000–F0FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

#### Field Descriptions

**Memory Type Address F0000–F0FFFh (MtrrFix4kF0xxxMemType)**—Bits 7–0. Memory type for physical address F0000–F0FFFh.

**Memory Type Address F1000–F1FFFh (MtrrFix4kF1xxxMemType)**—Bits 15–8. Memory type for physical address F1000–F1FFFh.

**Memory Type Address F2000–F2FFFh (MtrrFix4kF2xxxMemType)**—Bits 23–16. Memory type for physical address F2000–F2FFFh.

**Memory Type Address F3000–F3FFFh (MtrrFix4kF3xxxMemType)**—Bits 31–24. Memory type for physical address F3000–F3FFFh.

**Memory Type Address F40000–F4FFFh (MtrrFix4kF4xxxMemType)**—Bits 39–32. Memory type for physical address F4000–F4FFFh.

**Memory Type Address F5000–F5FFFh (MtrrFix4kF5xxxMemType)**—Bits 47–40. Memory type for physical address F5000–F5FFFh.

**Memory Type Address F6000–F6FFFh (MtrrFix4kF6xxxMemType)**—Bits 55–48. Memory type for physical address F6000–F6FFFh.

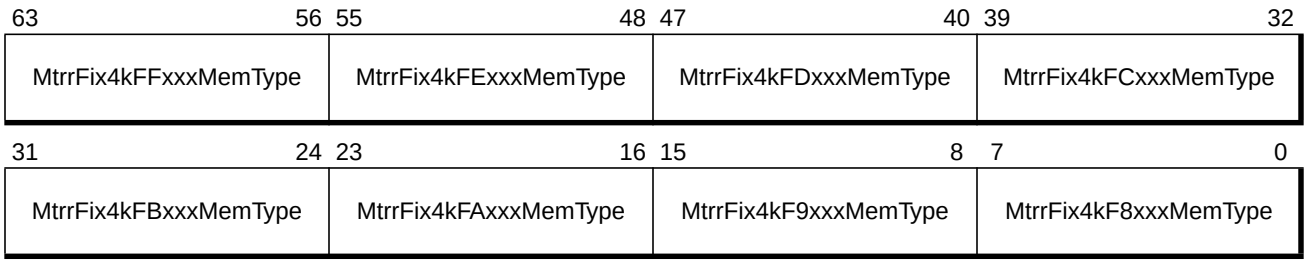
**Memory Type Address F7000–F7FFFh (MtrrFix4kF7xxxMemType)**—Bits 63–56. Memory type for physical address F7000–F7FFFh.

### 12.1.2.14 MTRRfix4K_F8000 Register

This register controls the memory types for physical memory addresses F8000–FFFFFh. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K_F8000 Register** **MSR 026Fh**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–56 | MtrrFix4kFFxxxMemType | Memory type address FF000–FFFFFh | R/W | U |
| 55–48 | MtrrFix4kFExxxMemType | Memory type address FE000–FEFFFh | R/W | U |
| 47–40 | MtrrFix4kFDxxxMemType | Memory type address FD000–FDFFFh | R/W | U |
| 39–32 | MtrrFix4kFCxxxMemType | Memory type address FC000–FCFFFh | R/W | U |
| 31–24 | MtrrFix4kFBxxxMemType | Memory type address FB000–FBFFFh | R/W | U |
| 23–16 | MtrrFix4kFAxxxMemType | Memory type address FA000–FAFFFh | R/W | U |
| 15–8 | MtrrFix4kF9xxxMemType | Memory type address F9000–F9FFFh | R/W | U |
| 7–0 | MtrrFix4kF8xxxMemType | Memory type address F8000–F8FFFh | R/W | U |

* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Memory Type Address F8000–F8FFFh (MtrrFix4kF8xxxMemType)**—Bits 7–0. Memory type for physical address F8000–F8FFFh.

**Memory Type Address F9000–F9FFFh (MtrrFix4kF9xxxMemType)**—Bits 15–8. Memory type for physical address F9000–F9FFFh.

**Memory Type Address FA000–FAFFFh (MtrrFix4kFAxxxMemType)**—Bits 23–16. Memory type for physical address FA000–FAFFFh.

**Memory Type Address FB000–FBFFFh (MtrrFix4kFBxxxMemType)**—Bits 31–24. Memory type for physical address FB000–FBFFFh.

**Memory Type Address FC0000–FCFFFh (MtrrFix4kFCxxxMemType)**—Bits 39–32. Memory type for physical address FC000–FCFFFh.

**Memory Type Address FD000–FDFFFh (MtrrFix4kFDxxxMemType)**—Bits 47–40. Memory type for physical address FD000–FDFFFh.

**Memory Type Address FE000–FEFFFh (MtrrFix4kFExxxMemType)**—Bits 55–48. Memory type for physical address FE000–FEFFFh.

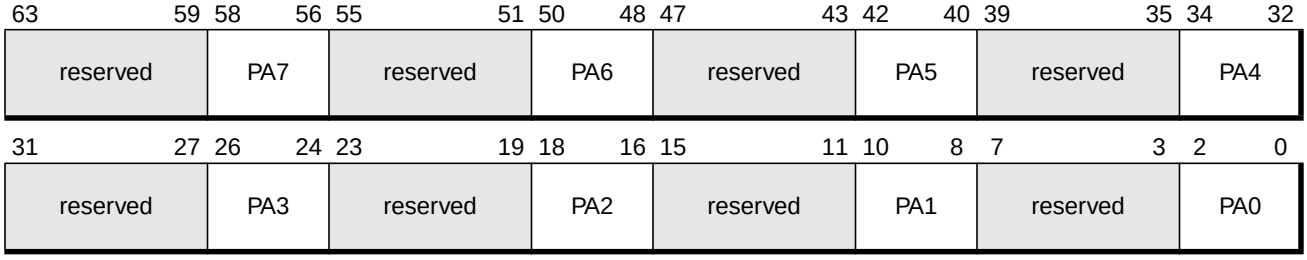
**Memory Type Address FF000–FFFFFh (MtrrFix4kFFxxxMemType)**—Bits 63–56. Memory type for physical address FF000–FFFFFh.

### 12.1.2.15 PAT Register

This register contains the eight page attribute fields used for specifying memory types for pages. See the "Page-Attribute Table Mechanism" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0). Setting any reserved bits will result in a #GP(0).

**PAT Register** **MSR 0277h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–59 | reserved | MBZ | | 0 |
| 58–56 | PA7 | Memory type for Page Attribute index 7 | R/W | 0 |
| 55–51 | reserved | MBZ | | 0 |
| 50–48 | PA6 | Memory type for Page Attribute index 6 | R/W | 7 |
| 47–43 | reserved | MBZ | | 0 |
| 42–40 | PA5 | Memory type for Page Attribute index 5 | R/W | 4 |
| 39–35 | reserved | MBZ | | 0 |
| 34–32 | PA4 | Memory type for Page Attribute index 4 | R/W | 6 |
| 31–27 | reserved | MBZ | | 0 |

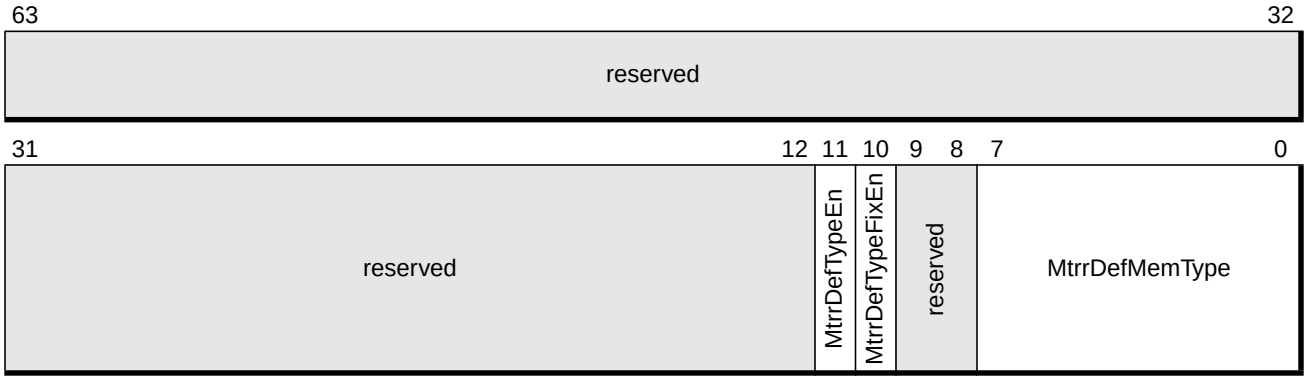
| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 26–24 | PA3 | Memory type for Page Attribute index 3 | R/W | 0 |
| 23–19 | reserved | MBZ | | 0 |
| 18–16 | PA2 | Memory type for Page Attribute index 2 | R/W | 7 |
| 15–11 | reserved | MBZ | | 0 |
| 10–8 | PA1 | Memory type for Page Attribute index 1 | R/W | 4 |
| 7–3 | reserved | MBZ | | 0 |
| 2–0 | PA0 | Memory type for Page Attribute index 0 | R/W | 6 |

### 12.1.2.16 MTRRdefType Register

This register enables the MTRRs and defines the default memory type for memory not within one of the MTRR ranges. See the "Memory Type Range Registers" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting MtrrDefMemType to an undefined memory type will result in a #GP(0). Setting any reserved bits will result in a #GP(0).

**MTRRdefType Register** **MSR 02FFh**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–12 | reserved | MBZ | | 0 |
| 11 | MtrrDefTypeEn | Enable MTRRs | R/W | 0 |
| 10 | MtrrDefTypeFixEn | Enable Fixed Range MTRRs | R/W | 0 |
| 9–8 | reserved | | | 0 |
| 7–0 | MtrrDefMemType | Default Memory Type | R/W | 0 |

* Memory types must be set to values consistent with system hardware.

#### Field Descriptions

**Default Memory Type (MtrrDefMemType)**—Bit 7–0.

**Enable Fixed Range MTRRs (MtrrDefTypeFixEn)**—Bit 10.

**Enable MTRRs (MtrrDefTypeEn)**—Bit 11.

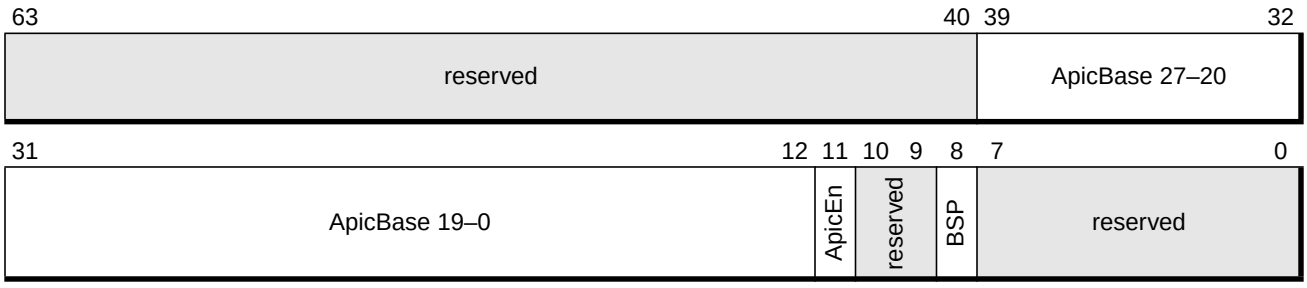
## 12.1.3 APIC Registers

### 12.1.3.1 APIC_BASE Register

This register enables APIC and defines the APIC base address. It also identifies the Boot Strap Processor (BSP).

Setting any reserved bits will result in a #GP(0).

**APIC_BASE Register** **MSR 001Bh**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–40 | reserved | MBZ | | 0 |
| 39–12 | ApicBase | APIC Base Address | R/W | 0x0FEE00 |
| 11 | ApicEn | Enable APIC | R/W | 0 |
| 10–9 | reserved | MBZ | | 0 |
| 8 | BSP | Boot Strap Processor | R/W | 1 if uniprocessor or bsp of multiprocessor system |
| 7–0 | reserved | MBZ | | 0 |

* APIC configuration must be consistent with system hardware.

#### Field Descriptions

**Boot Strap Processor (Bsp)**—Bit 8.

**Enable APIC (ApicEn)**—Bit 11.

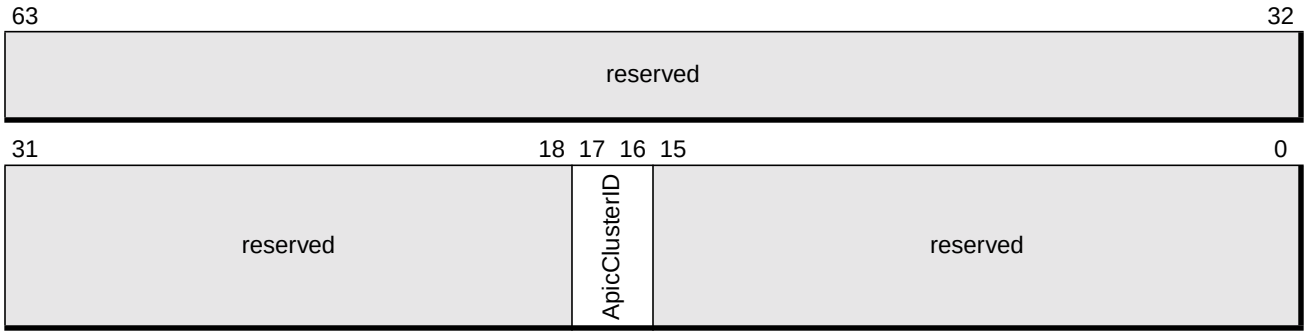
**APIC Base Address (ApicBase)**—Bits 39–12.

### 12.1.3.2 EBL_CR_POWERON Register

This read-only register contains the APIC cluster ID. See the "APIC Initialization" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Attempting to write to this register will result in a #GP(0).

### EBL_CR_POWERON Register — MSR 002Ah




| Bit | Mnemonic | Function | R/W |
|---|---|---|---|
| 63–18 | reserved | MBZ | |
| 17–16 | ApicClusterID | APIC Cluster ID | R |
| 15–0 | reserved | MBZ | |

#### Field Descriptions

**APIC Cluster ID (ApicClusterID)**—Bits 17–16.

## 12.1.4 Software Debug Registers

The AMD Athlon™ 64 and AMD Opteron™ processors incorporate extensive debug facilities. These include the following control, status, and control-transfer recording MSRs.

- DebugCtl Register (MSR 01D9h)—Provides additional debug controls over control-transfer recording and single stepping, as well as external-breakpoint reporting and trace messages.
- LastBranchFromIP Register (MSR 01DBh)—Loaded with the segment offset of the branch instruction.
- LastBranchToIP Register (MSR 01DCh)—Holds the target rIP of the last branch that occurred before an exception or interrupt.
- LastExceptionFromIP Register (MSR 01DDh)—Holds the source rIP of the last branch that occurred before the exception or interrupt.
- LastExceptionToIP Register (MSR 01DEh)—Holds the target rIP of the last branch that occurred before the exception or interrupt.

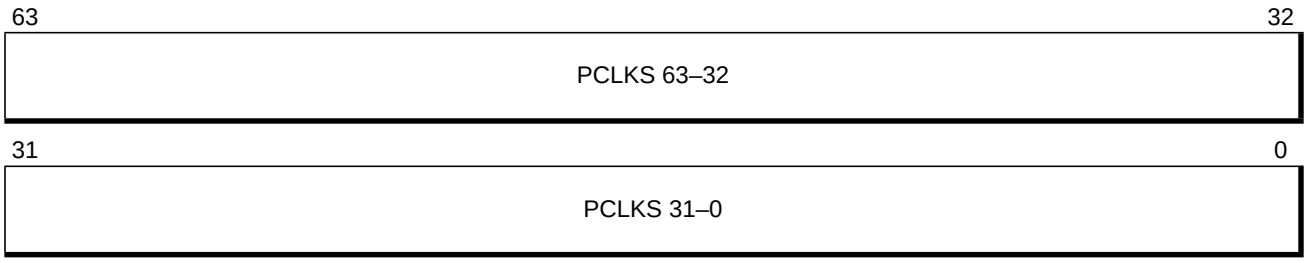
For more detailed information on the use of these MSRs, see the "Debug and Performance Resources" section in Volume 2 of the *AMD64 Architecture Programmer's Manual*.

### 12.1.5 Performance Monitoring Registers

#### 12.1.5.1 TSC Register

The time-stamp counter (TSC) register maintains a running count of the number of internal processor clock cycles executed after a reset. It is incremented by 1 on each internal processor clock. When the TSC overflows its 64-bit range, it wraps around to 0. See the "Time-Stamp Counter" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

**TSC Register** **MSR 0010h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–0 | PCLKS | Running Clock Cycle Count | R/W | 0 |

**Field Descriptions**

**Processor Clock Cycles (PCLKS)**—Bits 63–0. Running count of number of internal processor clock cycles.

## 12.2 AMD Athlon™ 64 processor and AMD Opteron™ Processor Model-Specific Registers

Table 57 is a listing of the model-specific registers supported by the AMD Athlon™ 64 processor and AMD Opteron™ processor, presented in ascending hexadecimal address order. Register descriptions follow the table, organized according to the following functions:

- Features (see page 286)
- Identification (see page 291)
- Memory typing (see page 292)
- I/O range registers (see page 293)
- System call extension registers (see page 294)
- Segmentation (see page 296)

- System management (see page 167)
- Power management (see page 298)

**Table 57. AMD Athlon™ 64 Processor and AMD Opteron™ Processor MSRs**

| Address | Register Name | Description |
|---|---|---|
| C000_0080h | EFER | page 286 |
| C000_0081h | STAR | page 294 |
| C000_0082h | LSTAR | page 295 |
| C000_0083h | CSTAR | page 295 |
| C000_0084h | SF_MASK | page 296 |
| C000_0100h | FS.Base | page 296 |
| C000_0101h | GS.Base | page 297 |
| C000_0102h | KernelGSbase | page 297 |
| C001_0000h–C001_0003h | PerfEvtSel*i* | page 246 |
| C001_0004h–C001_0007h | PerfCtr*i* | page 245 |
| C001_0010h | SYSCFG | page 286 |
| C001_0015h | HWCR | page 288 |
| C001_0016h, C001_0018h | IORRBase[1:0] | page 293 |
| C001_0017h, C001_0019h | IORRMask[1:0] | page 293 |
| C001_001Ah | TOP_MEM | page 292 |
| C001_001Dh | TOP_MEM2 | page 292 |
| C001_001Eh | MANID | page 291 |
| C001_001Fh | NB_CFG | page 290 |
| C001_0041h | FIDVID_CTL | page 298 |
| C001_0042h | FIDVID_STATUS | page 300 |
| C001_0050–C001_0053h | IOTRAP_ADDR*i* | page 302 |
| C001_0054h | IOTRAP_CTL | page 303 |
| C001_0111h | SMM_BASE | page 173 |
| C001_0112h | SMM_ADDR | page 178 |
| C001_0113h | SMM_MASK | page 177 |

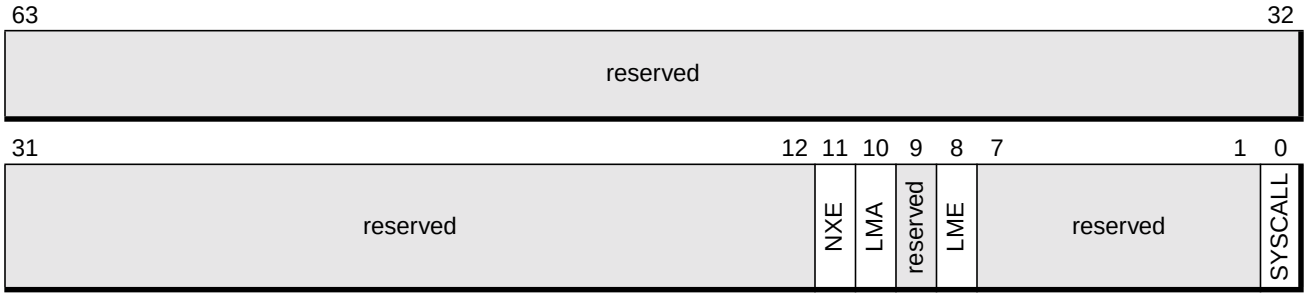
## 12.2.1 Feature Registers

### 12.2.1.1 EFER Register

This register controls which extended features are enabled. See the "Extended Feature Enable Register (EFER)" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

The LMA bit is a read-only status bit. When writing EFER, the processor will signal a #GP(0) if an attempt is made to change LMA from its previous value. BIOS should leave this register alone unless it needs to use one of the features enabled by this register.

**EFER Register** **MSR C000_0080h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–12 | reserved | MBZ | R/W | 0 |
| 11 | NXE | No-Execute Page Enable | R/W | 0 |
| 10 | LMA | Long Mode Active | R | 0 |
| 9 | reserved | MBZ | R | 0 |
| 8 | LME | Long Mode Enable | R/W | 0 |
| 7–1 | reserved | RAZ | R | 0 |
| 0 | SYSCALL | System Call Extension Enable | R/W | 0 |

#### Field Descriptions

**System Call Extension Enable (SCE)**—Bit 0. Enables the system call extension

**Long Mode Enable (LME)**—Bit 8. Enables the long mode feature

**Long Mode Active (LMA)**—Bit 10. Indicates the long mode feature is active

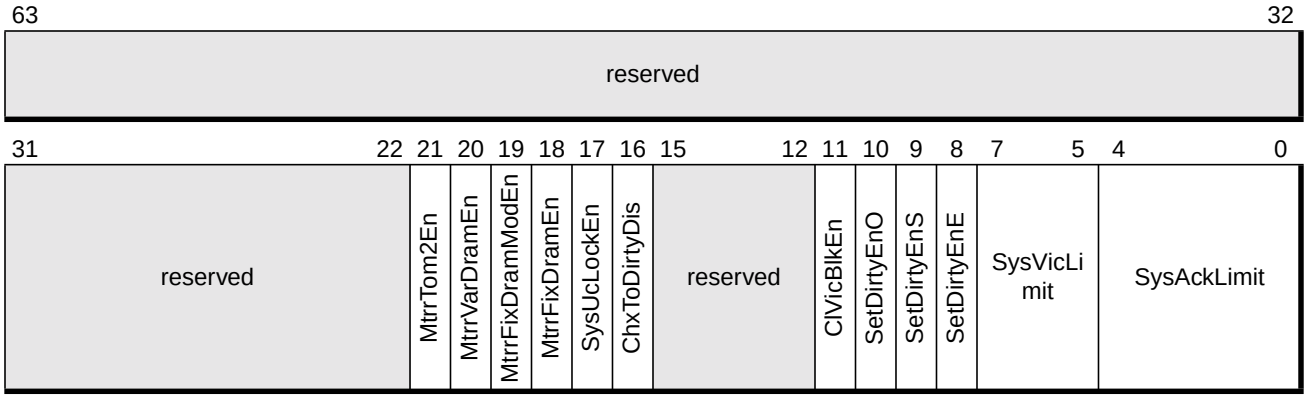
**No-Execute Page Enable (NXE)**—Bit 11. Enables the no-execute page feature

### 12.2.1.2 SYSCFG Register

This register controls the system configuration.

The MtrrFixDramModEn bit should be set to 1 during BIOS initalization of the fixed MTRRs, then cleared to 0 for operation.

## SYSCFG Register — MSR C001_0010h




| Bit | Mnemonic | Function | R/W | Reset | BIOS |
|---|---|---|---|---|---|
| 63–22 | reserved | RAZ | R | 0 | |
| 21 | MtrrTom2En | Top of Memory Address Register 2 Enable (reserved) | R/W | 0 | 0 |
| 20 | MtrrVarDramEn | Top of Memory Address Register and I/O Range Register Enable | R/W | 0 | 1 |
| 19 | MtrrFixDramModEn | RdDram and WrDram Bits Modification Enable | R/W | 0 | 0 |
| 18 | MtrrFixDramEn | Fixed RdDram and WrDram Attributes Enable | R/W | 0 | 1 |
| 17 | SysUcLockEn | System Interface Lock Command Enable | R/W | 1 | 1 |
| 16 | ChxToDirtyDis | Change to Dirty Command Disable | R/W | 0 | 0 |
| 15–12 | reserved | RAZ | R | 0 | |
| 11 | ClVicBlkEn | Revision B and earlier revisions: ClVicBlk System Interface Command Enable<br>Revision C: Reserved | R/W | 0 | 0 |
| 10 | SetDirtyEnO | CleanToDirty Command for O->M State Transition Enable | R/W | 1 | 1 |
| 9 | SetDirtyEnS | SharedToDirty Command for S->M State Transition Enable | R/W | 1 | 1 |
| 8 | SetDirtyEnE | SharedToDirty Command for E->M State Transition Enable | R/W | 0 | 0 |
| 7–5 | SysVicLimit | Outstanding Victim Bus Command Limit | R/W | 000b | 000b |
| 4–0 | SysAckLimit | Outstanding Bus Command Limit | R/W | 00001b | 00001b |

### Field Descriptions

**Outstanding Bus Command Limit (SysAckLimit)**—Bit 4–0. Limits maximum number of outstanding bus commands.

**Outstanding Victim Bus Command Limit (SysVicLimit)**—Bit 7–5. Limits maximum number of outstanding victim bus commands.

**SharedToDirty Command for E->M State Transition Enable (SetDirtyEnE)**—Bit 8. Enables SharedToDirty commands for E->M state transitions.

**SharedToDirty Command for S->M State Transition Enable (SetDirtyEnS)**—Bit 9. Enables SharedToDirty commands for S->M state transitions.

**CleanToDirty Command for O->M State Transition Enable (SetDirtyEnO)**—Bit 10. Enables CleanToDirty commands for O->M state transitions.

**ClVicBlk System Interface Command Enable (ClVicBlkEn)**—Bit 11. Enables ClVicBlk system interface command.

**Change to Dirty Command Disable (ChxToDirtyDis)**—Bit 16. Disables change to dirty commands, evicts line from DC instead.

**System Interface Lock Command Enable (SysUcLockEn)**—Bit 17. Enables lock commands on system interface.

**Fixed RdDram and WrDram Attributes Enable (MtrrFixDramEn)**—Bit 18. Enables fixed MTRR RdDram and WrDram attributes.

**RdDram and WrDram Bits Modification Enable (MtrrFixDramModEn)**—Bit 19. Enables modification of RdDram and WrDram bits in fixed MTRRS.

**Top of Memory Address Register and I/O Range Register Enable (MtrrVarDramEn)**—Bit 20. Enables use of top of memory address register and the I/O range registers.

**Top of Memory Address Register 2 Enable (MtrrTom2En)**—Bit 21. Enables use of top of memory address register 2 (reserved).

#### 12.2.1.3 HWCR Register

This register controls the hardware configuration.

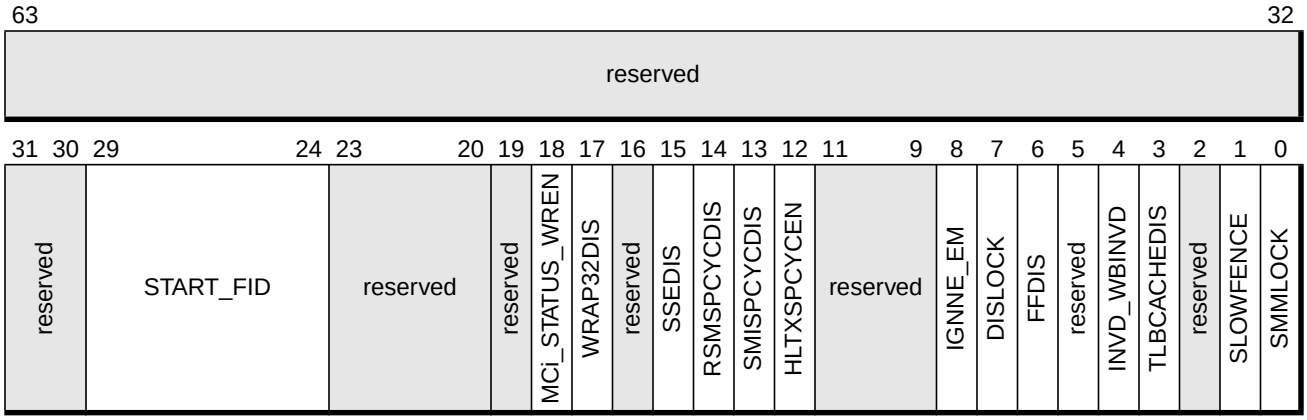
Operating systems that maintain page tables in uncacheable memory (UC memory type) must set the TLBCACHEDIS bit to insure proper operation.

BIOS and SMM handlers may use the WRAP32DIS bit to access physical memory above 4 Gbytes without switching into 64-bit mode. By setting WRAP32DIS to a 1 in conjunction with setting the expanded FS or GS base registers, BIOS can size all of physical memory from legacy mode. To do so, BIOS would write a >32 bit base to the FS or GS base register using WRMSR. Then it would address 2 Gbytes from one of those bases using normal memory reference instructions with a FS or GS override prefix. However, the INVLPG, FST, and SSE store instructions generate 32-bit addresses in legacy mode, regardless of the state of WRAP32DIS.

The MCi_STATUS_WREN bit can be used to debug Machine Check handlers. When MCi_STATUS_WREN bit is set, privileged software can write non-zero values to the Machine Check MSRs (MSRs 017Ah, 0401h, 0405h, 0409h, 040Dh, 0411h, 0402h, 0406h, 040Ah, 040Eh, 0412h) without generating exceptions, and then simulate a machine check using the "int 18" instruction.

Setting a reserved bit in these MSRs does not generate an exception when this mode is enabled. However, setting a reserved bit may result in undefined behavior.

## HWCR Register — MSR C001_0015h




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–32 | reserved | RAZ | R | 0 |
| 31–30 | reserved | SBZ | R/W | 0 |
| 29–24 | START_FID | Startup FID Status | R | 0 |
| 23–20 | reserved | RAZ | R | 0 |
| 19 | reserved | SBZ | R/W | 0 |
| 18 | MCi_STATUS_WREN | MCi Status Write Enable | R/W | 0 |
| 17 | WRAP32DIS | 32-bit Address Wrap Disable | R/W | 0 |
| 16 | reserved | | | 0 |
| 15 | SSEDIS | SSE Instructions Disable | R/W | 0 |
| 14 | RSMSPCYCDIS | Special Bus Cycle On RSM Disable | R/W | 0 |
| 13 | SMISPCYCDIS | Special Bus Cycle On SMI Disable | R/W | 0 |
| 12 | HLTXSPCYCEN | Enable Special Bus Cycle On Exit From HLT | R/W | 0 |
| 11–9 | reserved | SBZ | R/W | 0 |
| 8 | IGNNE_EM | IGNNE Port Emulation Enable | R/W | 0 |
| 7 | DISLOCK | Disable x86 LOCK prefix functionality | R/W | 0 |
| 6 | FFDIS | TLB Flush Filter Disable | R/W | 0 |
| 5 | reserved | SBZ | R/W | 0 |
| 4 | INVD_WBINVD | INVD to WBINVD Conversion | R/W | 0 |
| 3 | TLBCACHEDIS | Cacheable Memory Disable | R/W | 0 |
| 2 | reserved | SBZ | R/W | 0 |
| 1 | SLOWFENCE | Slow SFENCE Enable | R/W | 0 |
| 0 | SMMLOCK | SMM Code Lock | R/W | 0 |

### Field Descriptions

**SMM Code Lock (SMMLOCK)**—Bit 0. Locks SMM code in TSeg range by making it read-only.

**Slow SFENCE Enable (SLOWFENCE)**—Bit 1. Enable slow sfence.

**Cachable Memory Disable (TLBCACHEDIS)**—Bit 3. Disable performance improvement that assumes that the PML4, PDP, PDE and PTE entries are in cacheable memory. If page tables are uncachable, TLBCACHEDIS must be set to ensure correct functionality.

**INVD to WBINVD Conversion (INVD_WBINVD)**—Bit 4. Convert INVD to WBINVD.

**TLB Flush Filter Disable (FFDIS)**—Bit 6. Disable TLB flush filter.

**Disable LOCK (DISLOCK)**—Bit 7. Disable x86 LOCK prefix functionality.

**IGNNE Port Emulation Enable (IGNNE_EM)**—Bit 8. Enable emulation of IGNNE port.

**Special Bus Cycle On HLT Exit Enable (HLTXSPCYCEN)**—Bit 12. Enables special bus cycle generation on exit from HLT. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

**Special Bus Cycle On SMI Disable (SMISPCYCDIS)**—Bit 13. Disables special bus cycle on SMI.

**Special Bus Cycle On RSM Disable (RSMSPCYCDIS)**—Bit 14. Disables special bus cycle on RSM.

**SSE Instructions Disable (SSEDIS)**—Bit 15. Disables SSE instructions.

**32-bit Address Wrap Disable (WRAP32DIS**)—Bit 17. Disable 32-bit address wrapping.

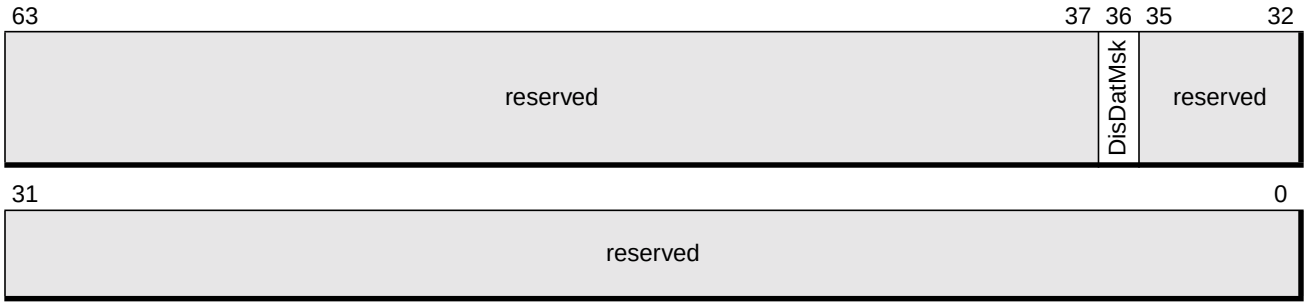
**MCi Status Write Enable (MCi_STATUS_WREN**)—Bit 18. When set, writes by software to Machine Check Error Status MSRs, Machine Check Error Address MSRs, and Machine Check MISC MSRs (if implemented) do not cause general protection faults. Such writes update all implemented bits in these registers. When clear, writing a non-zero pattern to these registers causes a general protection fault. See the description above for more details.

**Startup FID Status (START_FID)**—Bits 29–24. Status of the startup FID.

### 12.2.1.4 NB_CFG Register

Software must perform a read-modify-write to this register to change its value.

**NB_CFG Register** **MSR C001_001Fh**



| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–37 | reserved | | R/W | 0 |
| 36 | DisDatMsk | Disable Data Mask | R/W | 0 |
| 35–0 | reserved | | R/W | 0 |

### Field Descriptions

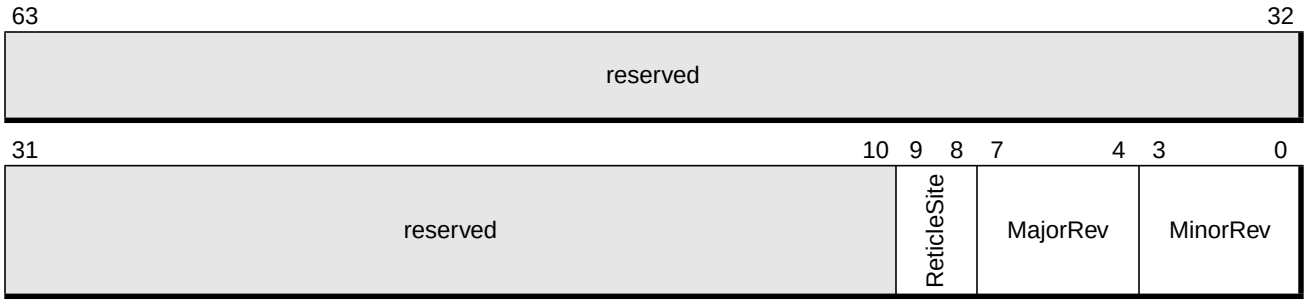
**Disable Data Mask (DisDatMsk)**—Bits 36. Disables DRAM data masking function. For all sized write requests a DRAM read is performed before writing the data. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this field.

## 12.2.2 Identification Registers

### 12.2.2.1 MANID Register

This status register holds the mask manufacturing identification number.

**MANID Register** **MSR C001_001Eh**




| Bit | Mnemonic | Function | R/W |
|---|---|---|---|
| 63–10 | reserved | | |
| 9–8 | ReticleSite | Reticle site | R |
| 7–4 | MajorRev | Major mask set revision number | R |
| 3–0 | MinorRev | Minor mask set revision number | R |

### Field Descriptions

**Minor Mask Set Revision Number (MinorRev)**—Bits 3–0.

**Major Mask Set Revision Number (MajorRev)**—Bits 7–4.

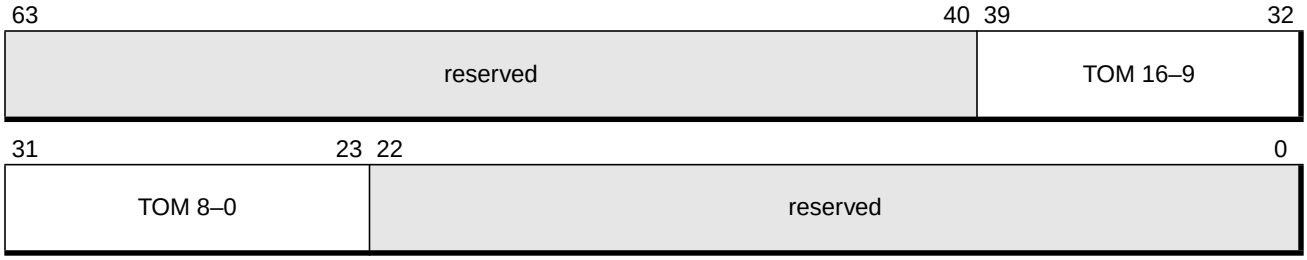
**Reticle Site (ReticleSite)**—Bits 9–8.

## 12.2.3 Memory Typing Registers

### 12.2.3.1 TOP_MEM Register

This register holds the address of the top of memory. When enabled, this address indicates the first byte of I/O above DRAM.

**TOP_MEM Register** **MSR C001_001Ah**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–40 | reserved | RAZ | | |
| 39–23 | TOM | Top of Memory | R/W | |
| 22–0 | reserved | RAZ | | |

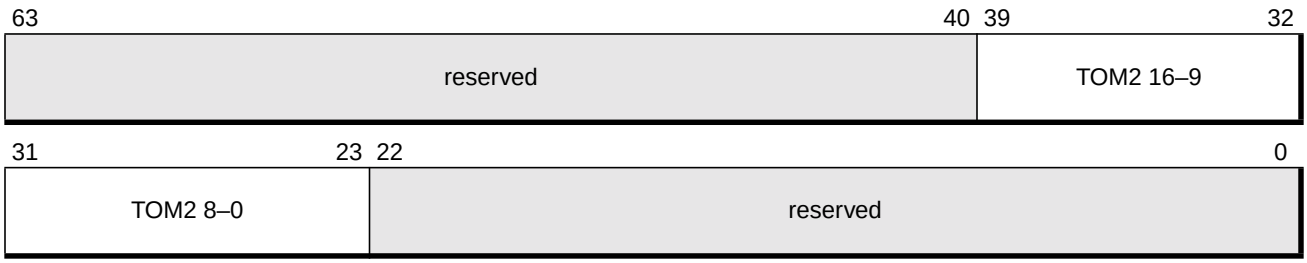
#### Field Descriptions

**Top of Memory (TOM)**—Bits 39–23. Address of top of memory (8-Mbyte granularity).

### 12.2.3.2 TOP_MEM2 Register

This register holds the second top of memory address. When enabled by setting the MtrrTom2En bit in the SYSCFG register, this address indicates the first byte of I/O above a second allocation of DRAM that starts at 4 Gbytes. Hence, the address in TOP_MEM2 should be set above 4 Gbytes.

**TOP_MEM2 Register** **MSR C001_001Dh**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–40 | reserved | RAZ | | |

| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 39–23 | TOM2 | Second Top of Memory | R/W | |
| 22–0 | reserved | RAZ | | |

### Field Descriptions

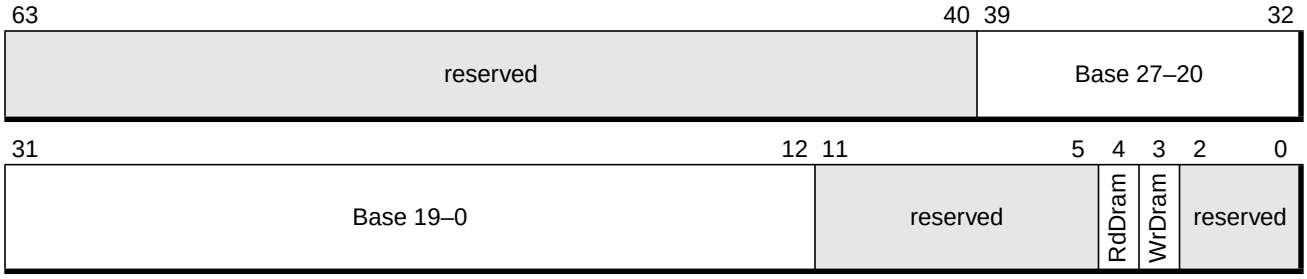
**Second Top of Memory (TOM2)**—Bits 39–23. Address of second top of memory (8-Mbyte granularity).

## 12.2.4 I/O Range Registers

### 12.2.4.1 IORRBase*i* Registers

These registers hold the bases of the variable I/O ranges.

**IORRBase0–1 Registers** **MSRs C001_0016h, C001_0018h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–40 | reserved | RAZ | | |
| 39–12 | Base | Base address | R/W | |
| 11–5 | reserved | RAZ | | |
| 4 | RdDram | Read from DRAM | R/W | |
| 3 | WrDram | Write to DRAM | R/W | |
| 2–0 | reserved | RAZ | | |

### Field Descriptions

**Write to DRAM (WrDram)**—Bit 3. If set, stores write to DRAM, otherwise I/O for this range.

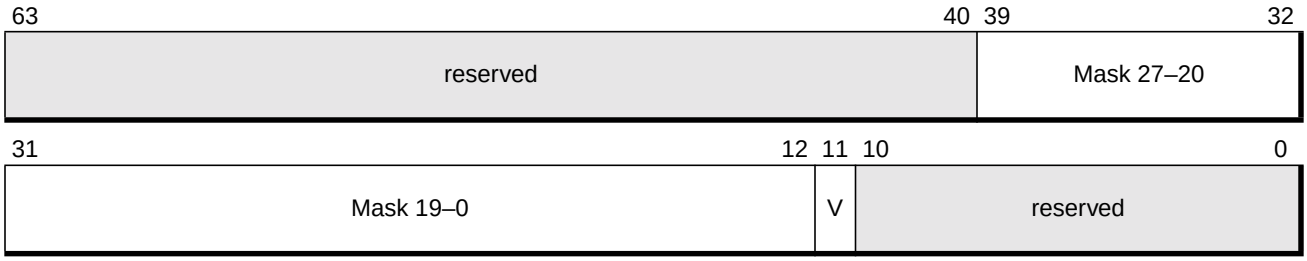
**Read from DRAM (RdDram)**—Bit 4. If set, fetches read from DRAM, otherwise from I/O for this range.

**Base Address (Base)**—Bits 39–12. Base address for this range.

### 12.2.4.2 IORRMask*i* Registers

These registers hold the masks of the variable I/O ranges.

### IORRMask0–1 Registers — MSRs C001_0017h, C001_0019h




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–40 | reserved | RAZ | | |
| 39–12 | Mask | Address mask | R/W | |
| 11 | V | Enables variable I/O range registers | R/W | |
| 10–0 | reserved | RAZ | | |

#### Field Descriptions

**Variable I/O Range (V)**—Bit 11. Enables variable I/O range register.

**Address Mask (Mask)**—Bits 39–12. Address mask.

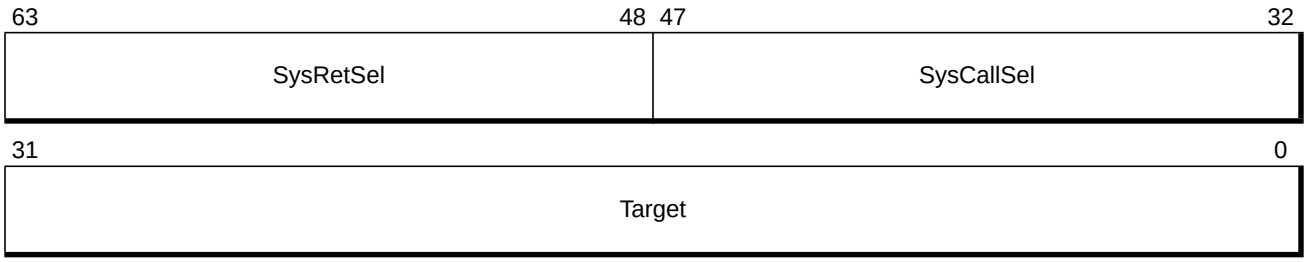
## 12.2.5 System Call Extension Registers

The system call extension is enabled by setting bit 0 in the EFER register. This feature adds the SYSCALL and SYSRET instructions which can be used in flat addressed operating systems as low latency system calls and returns. See the "SYSCALL and SYSRET" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

### 12.2.5.1 STAR Register

This register holds the target address used by the SYSCALL instruction and the code and stack segment selector bases used by the SYSCALL and SYSRET instructions.

### STAR Register — MSR C000_0081h



| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–48 | SysRetSel | SYSRET CS and SS | R/W | |
| 47–32 | SysCallSel | SYSCALL CS and SS | R/W | |
| 31–0 | Target | SYSCALL target address | R/W | |

### Field Descriptions

**SYSCALL Target Address (Target)**—Bits 31–0.

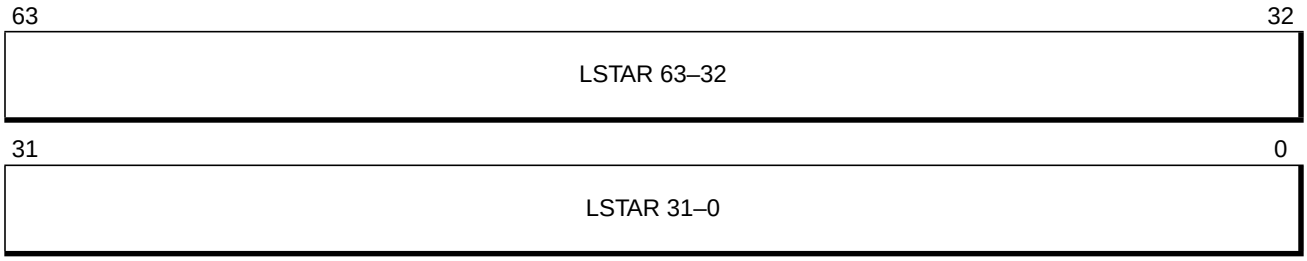
**SYSCALL CS and SS (SysCallSel)**—Bits 47–32.

**SYSRET CS and SS (SysRetSel)**—Bits 63–48.

### 12.2.5.2 LSTAR Register

The address stored in this register must be in canonical form (if not canonical, a #GP fault will occur).

**LSTAR Register** **MSR C000_0082h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–0 | LSTAR | Long Mode Target Address | R/W | |

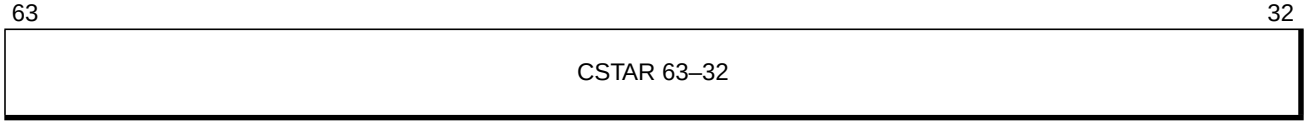
### Field Descriptions

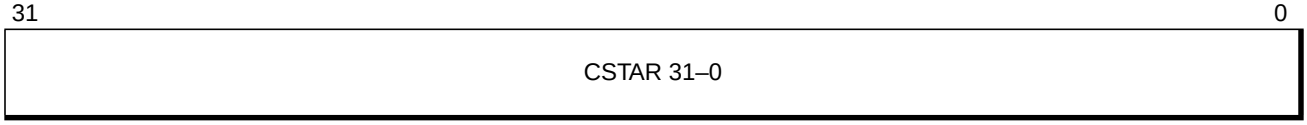
**Long Mode Target Address (LSTAR)**—Bits 63–0. Target address for 64-bit mode calling programs.

### 12.2.5.3 CSTAR Register

The address stored in this register must be in canonical form (if not canonical, a #GP fault fill occur).

**CSTAR Register** **MSR C000_0083h**






| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–0 | CSTAR | Compatibility mode target address | R/W | |

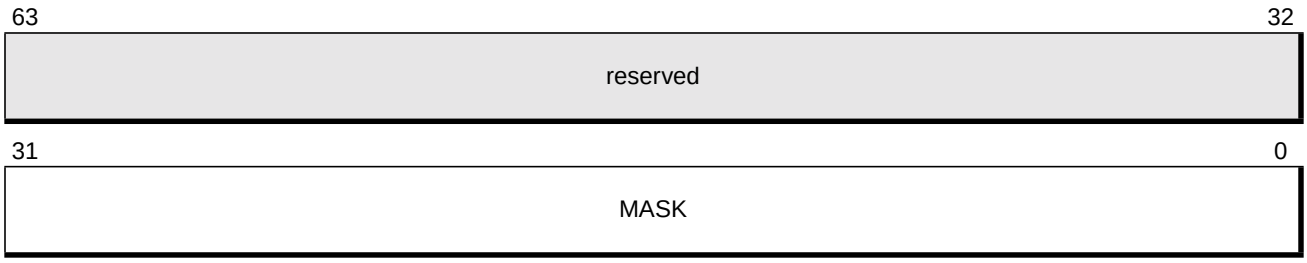
### Field Descriptions

**Compatibility Mode Target Address (CSTAR)**—Bits 63–0. Target address for compatibility mode calling programs.

#### 12.2.5.4 SF_MASK Register

This register holds the EFLAGS mask used by the SYSCALL instruction. Each one in this mask will clear the corresponding EFLAGS bit when executing the SYSCALL instruction.

**SF_MASK Register** **MSR C000_0084h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–32 | reserved | RAZ | | |
| 31–0 | MASK | SYSCALL Flag Mask | R/W | |

### Field Descriptions

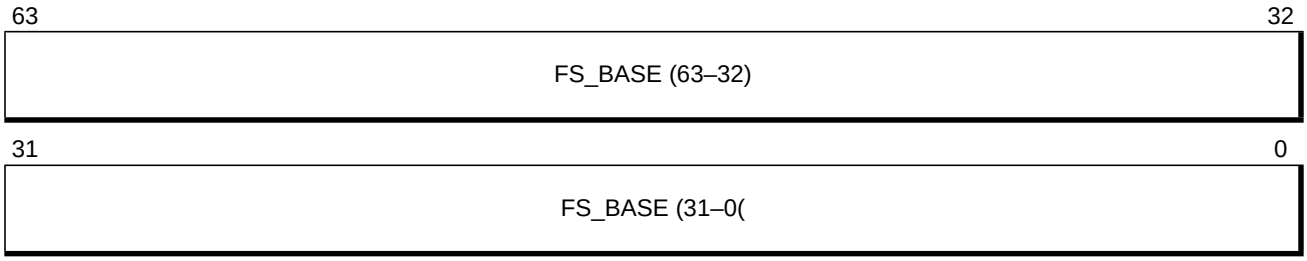
**SYSCALL Flag Mask (MASK)**—Bits 31–0.

## 12.2.6 Segmentation Registers

#### 12.2.6.1 FS.Base Register

This register provides access to the expanded 64 bit FS segment base. The address stored in this register must be in canonical form (if not canonical, a #GP fault fill occur).

### FS.Base Register — MSR C000_0100h




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–0 | FS_BASE | Expanded FS segment base | R/W | |

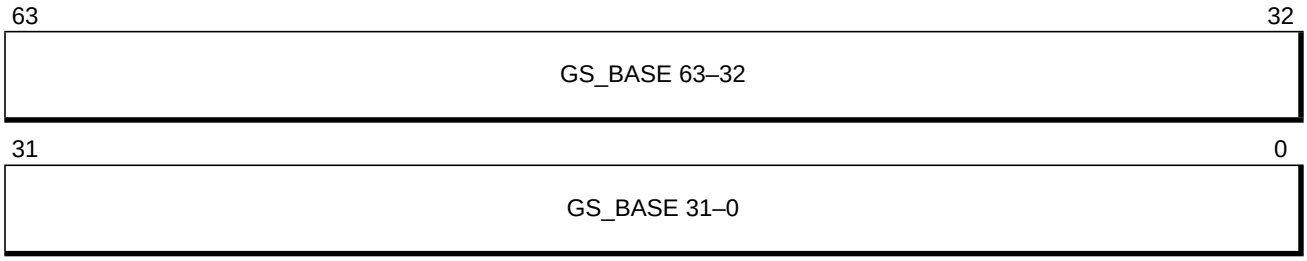
### Field Descriptions

**Expanded FS Segment Base (FS_BASE)**—Bits 63–0.

### 12.2.6.2 GS.Base Register

This register provides access to the expanded 64 bit GS segment base. The address stored in this register must be in canonical form (if not canonical, a #GP fault fill occur).

### GS.Base Register — MSR C000_0101h




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–0 | GS_BASE | Expanded GS segment base | R/W | |

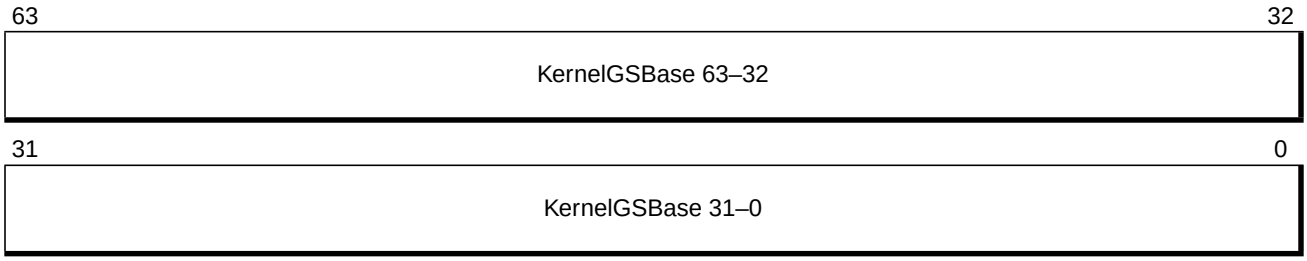
### Field Descriptions

**Expanded GS Segment Base (GS_BASE)**—Bits 63–0.

### 12.2.6.3 KernelGSbase Register

This register holds the kernel data structure pointer which can be swapped with the GS_BASE register using the new 64-bit mode instruction SwapGS. See the "SWAPGS Instruction" section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information. The address stored in this register must be in canonical form (if not canonical, a #GP fault will occur).

### KernelGSbase Register — MSR C000_0102h




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–0 | KernelGSBase | Kernel data structure pointer | R/W | |

### Field Descriptions

**Kernel Data Structure Pointer (KernelGSBase)**—Bits 63–0.

## 12.2.7 Power Management Registers

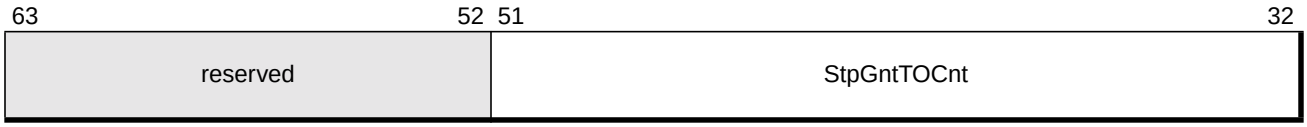
Mobile processors contain AMD PowerNow!™ technology hardware that allows the processor operating voltage and frequency to by dynamically controlled for power management purposes. Use CPUID function 8000_0007h (Get Advanced Power Management Feature Flags) to determine the thermal and power management capabilities of the processor. Refer to "Processor Performance States" on page 219 regarding the use of the FIDVID_STATUS and FIDVID_CTL MSRs.

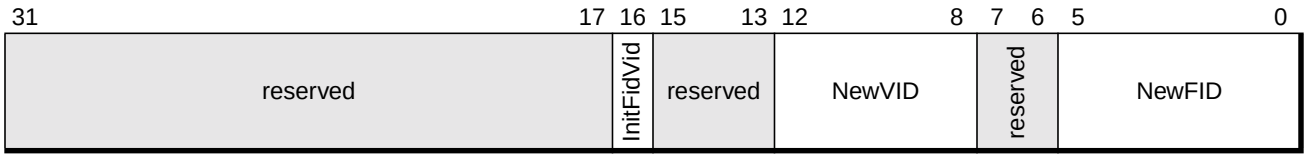
### 12.2.7.1 FIDVID_CTL Register

This register holds configuration information relating to power management control. Accessing this register is allowed if the device supports either FID control or VID control as indicated by CPUID. If neither FID control nor VID control is indicated, accessing the FIDVID_CTL register will cause a GP# fault.

This register is used to specify new Frequency ID (FID) and/or Voltage ID (VID) codes to which to change on the next FID/VID change. This register is also used to control how long to wait following a FID/VID change before the PLL and voltage regulator are stable at the new FID/VID. This register is persistent through a warm reset and is initialized to 0 on a cold reset.

### FIDVID_CTL Register — MSR C001_0041h






| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–52 | reserved | MBZ | | 0 |
| 51–32 | StpGntTOCnt | Stop Grant Time-Out Count | R/W | 0 |
| 31–17 | reserved | MBZ | | 0 |
| 16 | InitFidVid | Initiate FID/VID Change | W | 0 |
| 15–13 | reserved | MBZ | | 0 |
| 12–8 | NewVID | New VID | R/W | 0 |
| 7–6 | reserved | MBZ | R | 0 |
| 5–0 | NewFID | New FID | R/W | 0 |

## Field Descriptions

**Stop Grant Time-Out Count (StpGntTOCnt)**—Bits 51-32. This field carries a count of system clock cycles (5 ns) that must elapse from the time a new FID is applied until the time that the PLL is stable at the new FID.

**Initiate FID/VID Change (InitFidVid)**—Bits 16. Writing this bit to a 1 initiates a FID/VID change. In a multiprocessor system, only the bootstrap processor sees this bit written as a 1. Writing this bit to a 1 initiates the FID change special bus cycle; in Revision B and earlier revisions, the VID change special bus cycle was also initiated. This is a write-only bit and always reads as 0. The status of the resulting FID/VID change can be determined from the FIDVID_STATUS register.

**New VID (NewVID)**—Bits 12–8. This field is the new VID to transition to. If an attempt is made to write a NewVID value that corresponds to a voltage greater than the voltage that MaxVID corresponds to in the FIDVID_STATUS register then the MaxVID value is written instead. See Table 48 on page 235 for VID values and their corresponding voltages.

**New FID (NewFID)**—Bits 5–0. This field is the new FID to transition to. If an attempt is made to write a NewFID value greater than MaxFID in the FIDVID_STATUS register then the MaxFID value is written instead. See Table 58 for FID code translations.

**Table 58. FID Code Translations**

| Reference Clock Multiplier | 6-Bit FID Code | Reference Clock Multiplier | 6-Bit FID Code |
|---|---|---|---|
| 4x | 00_0000b | 15x | 01_0110b |
| 5x | 00_0010b | 16x | 01_1000b |
| 6x | 00_0100b | 17x | 01_1010b |

**Table 58. FID Code Translations (Continued)**

| Reference Clock Multiplier | 6-Bit FID Code | Reference Clock Multiplier | 6-Bit FID Code |
|---|---|---|---|
| 7x | 00_0110b | 18x | 01_1100b |
| 8x | 00_1000b | 19X | 01_1110b |
| 9x | 00_1010b | 20X | 10_0000b |
| 10x | 00_1100b | 21X | 10_0010b |
| 11x | 00_1110b | 22x | 10_0100b |
| 12x | 01_0000b | 23x | 10_0110b |
| 13x | 01_0010b | 24x | 10_1000b |
| 14x | 01_0100b | 25x | 10_1010b |

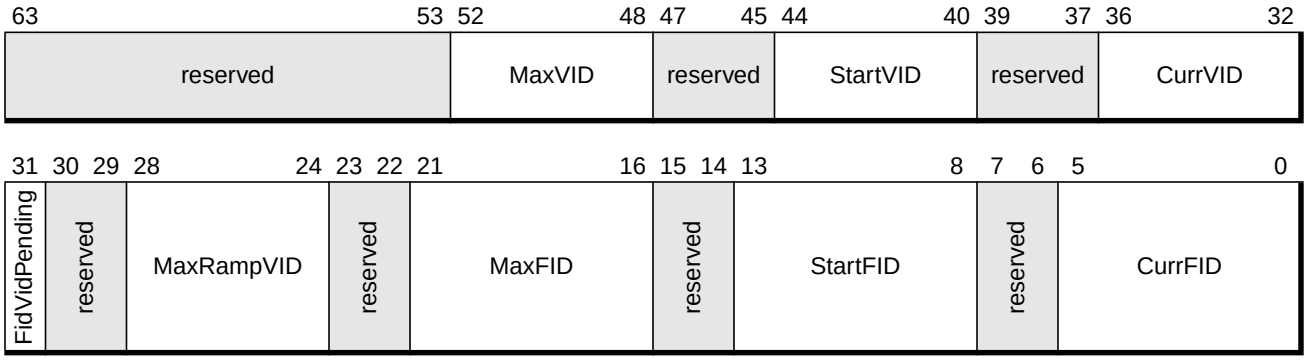
***Note:*** *Encodings not listed are reserved. Using reserved encodings will cause unpredictable behavior.*

### 12.2.7.2 FIDVID_STATUS Register

This register holds status of the current FID, the current VID, pending changes, and maximum values. Accessing this register is allowed if the device supports either FID control or VID control as indicated via CPUID. If neither FID control nor VID control is indicated, accessing the FIDVID_STATUS register will cause a GP# fault.

See Table 58 on page 299 for a complete list of FID values and their corresponding reference clock multiplier values. See Table 48 on page 235 for VID values and their corresponding voltages.

**FIDVID_STATUS Register** **MSR C001_0042h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–53 | reserved | RAZ | R | 0 |
| 52–48 | MaxVID | Max VID | R | 0 |
| 47–45 | reserved | RAZ | R | 0 |
| 44–40 | StartVID | Startup VID | R | 0 |
| 39–37 | reserved | RAZ | R | 0 |

| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 36–32 | CurrVID | Current VID | R | 0 |
| 31 | FidVidPending | FID/VID Change Pending | R | 0 |
| 30–29 | reserved | RAZ | R | 0 |
| 28–24 | MaxRampVID | Max Ramp VID | R | 0 |
| 23–22 | reserved | RAZ | R | 0 |
| 21–16 | MaxFID | Max FID | R | 0 |
| 15–14 | reserved | RAZ | R | 0 |
| 13–8 | StartFID | Startup FID | R | 0 |
| 7–6 | reserved | RAZ | R | 0 |
| 5–0 | CurrFID | Current FID | R | 0 |

### Field Descriptions

**Max VID (MaxVID)**—Bits 52–48. This field indicates the VID code associated with the maximum voltage software can select.

**Startup VID (StartVID)**—Bits 44–40. This field indicates the startup VID.

**Current VID (CurrVID)**—Bits 36–32. This field indicates the current VID.

**FID/VID Change Pending (FidVidPending)**—Bit 31. This bit indicates that a FID/VID change is pending. It is set to 1 by hardware when the InitFidVid bit is set in the FIDVID_CTL register. It is cleared by hardware when the FID/VID change has completed.

**Max Ramp VID (MaxRampVID)**—Bit 28–24. This field indicates the max Ramp VID.

**Max FID (MaxFID)**—Bit 21–16. This field indicates the max FID.

**Startup FID (StartFID)**—Bit 13–8. This field indicates the startup FID.

**Current FID (CurrFID)**—Bit 5–0. This field indicates the current FID.

## 12.2.8 IO and Configuration Space Trapping to SMI

IO and configuration space trapping to SMI is a mechanism for executing SMI handler if a specific IO access to one of the specified addresses is detected. Access address and access type checking is done before IO instruction execution. If the access address and access type match one of the specified IO address and access types, the SMI handler is executed, the IO instruction is not executed, and a breakpoint set on the IO instruction is not taken. The IO instruction can be executed inside the SMI handler. This mechanism has higher priority than the SMI# interrupt. If the trap is not taken, check for SMI# interrupt is done after the IO instruction has been executed.

Configuration accesses are special IO accesses. An IO access is defined as a configuration access when IO instruction address bits 31-0 are CFCh, CFDh, CFEh, or CFFh. The access address for a configuration space access is the current value of doubleword IO register CF8h. The access address

for an IO access that is not a configuration access is equivalent to the IO instruction address, bits 31–0.

The access address is compared with SmiAddr, and the instruction access type is compared with the enabled access types defined by ConfigSMI, SmiOnRdEn, and SmiOnWrEn. Access address bits 23-0 can be masked with SmiMask. Fields SmiAddr, SmiMask, ConfigSMI, SmiOnRdEn, and SmiOnWrEn are defined in IOTRAP_ADDR*i* registers (i = 0,1,2,3).

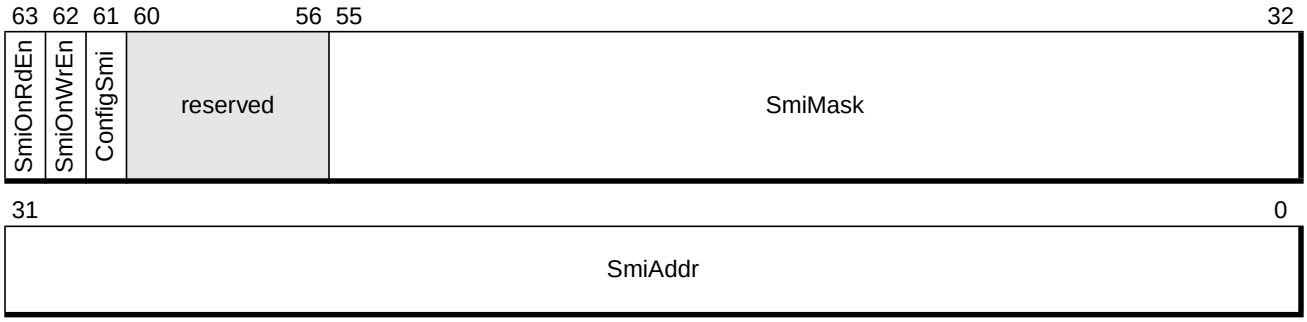
An SMI handler executed as a result of IO and configuration space trapping will by default only be detected by the processor executing the IO instruction. The user has the ability to enable an SMI special bus cycle and RSM special bus cycle generation.

IO and configuration space trapping to SMI applies only to single IO instructions; it does not apply to string and REP IO instructions.

### 12.2.8.1 IOTRAP_ADDR*i* Registers

See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this register.

**IOTRAP_ADDR*i* Registers** **MSRs C001_0050h, C001_0051h, C001_0052h, C001_0053h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63 | SmiOnRdEn | Enable SMI on IO Read | R/W | 0 |
| 62 | SmiOnWrEn | Enable SMI on IO Write | R/W | 0 |
| 61 | ConfigSmi | Configuration Space SMI | R/W | 0 |
| 60-56 | reserved | RAZ | R | |
| 55–32 | SmiMask | SMI Mask | R/W | 0 |
| 31–0 | SmiAddr | SMI Address | R/W | 0 |

#### Field Descriptions

**Enable SMI on IO Read (SmiOnRdEn)**—Bits 63. Enables SMI generation on a read access.

**Enable SMI on IO Write (SmiOnWrEn)**—Bit 62. Enables SMI generation on a write access.

**Configuration Space SMI (ConfigSmi)**—Bit 61. 1 = configuration access (see "IO and Configuration Space Trapping to SMI" on page 301 for more information on configuration access detection); 0 = IO access different than configuration access.

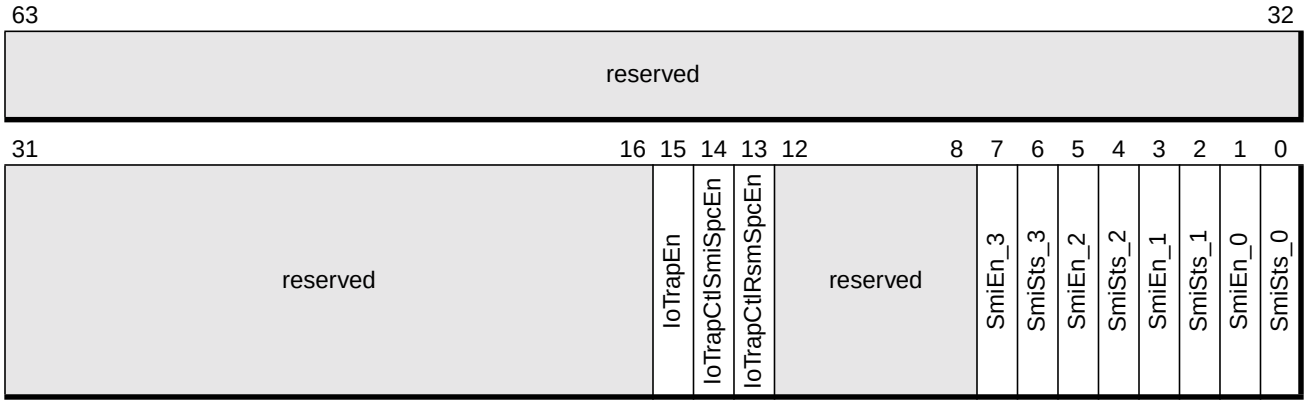
**SMI Mask (SmiMask)**—Bits 55–32. SMI IO trap mask. 0 = mask address bit; 1 = do not mask address bit.

**SMI Address (SmiAddr)**—Bits 31–0. SMI IO trap address.

### 12.2.8.2 IOTRAP_CTL Register

See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ processors" on page 19 for revision information about this register.

**IOTRAP_CTL Register** **MSR C001_0054h**




| Bit | Mnemonic | Function | R/W | Reset |
|---|---|---|---|---|
| 63–16 | reserved | RAZ | R | |
| 15 | IoTrapEn | IO Trap Enable | R/W | 0 |
| 14 | IoTrapCtlSmiSpcEn | IO Trap Control SMI Special Cycle Enable | R/W | 0 |
| 13 | IoTrapCtlRsmSpcEn | IO Trap Control RSM Special Cycle Enable | R/W | 0 |
| 12-8 | reserved | RAZ | R | |
| 7 | SmiEn_3 | SMI Enable 3 | R/W | 0 |
| 6 | SmiSts_3 | SMI Status 3 | R/W | 0 |
| 5 | SmiEn_2 | SMI Enable 2 | R/W | 0 |
| 4 | SmiSts_2 | SMI Status 2 | R/W | 0 |
| 3 | SmiEn_1 | SMI Enable 1 | R/W | 0 |
| 2 | SmiSts_1 | SMI Status 1 | R/W | 0 |
| 1 | SmiEn_0 | SMI Enable 0 | R/W | 0 |
| 0 | SmiSts_0 | SMI Status 0 | R/W | 0 |

#### Field Descriptions

**IO Trap Enable (IoTrapEn)**—Bit 15. Enable IO and configuration space trapping.

**IO Trap Control SMI Special Cycle Enable (IoTrapCtlSmiSpcEn)**—Bit 14. Enable SMI special bus cycle generation when SMI handler is entered as a result of IO and configuration space trapping.

**IO Trap Control RSM Special Cycle Enable (IoTrapCtlRsmSpcEn)**—Bit 13. Enable RSM special bus cycle generation when SMI handler is entered as a result of IO and configuration space trapping.

**SMI Enable 3 (SmiEn_3)**—Bit 7. Enable SMI generation if IO access matches access specified in MSR C001_0053h.

**SMI Status 3 (SmiSts_3)**—Bit 6. Set if IO access matched access specified in MSR C001_0053h.

**SMI Enable 2 (SmiEn_2)**—Bit 5. Enable SMI generation if IO access matches access specified in MSR C001_0052h.

**SMI Status 2 (SmiSts_2)**—Bit 4. Set if IO access matched access specified in MSR C001_0052h.

**SMI Enable 1 (SmiEn_1)**—Bit 3. Enable SMI generation if IO access matches access specified in MSR C001_0051h.

**SMI Status 1 (SmiSts_1)**—Bit 2. Set if IO access matched access specified in MSR C001_0051h.

**SMI Enable 0 (SmiEn_0)**—Bit 1. Enable SMI generation if IO access matches access specified in MSR C001_0050h.

**SMI Status 0 (SmiSts_0)**—Bit 0. Set if IO access matched access specified in MSR C001_0050h.

# Glossary

**128-bit media instructions.** Instructions that use the 128-bit XMM registers. These are a combination of SSE and SSE2 instruction sets.

**64-bit media instructions.** Instructions that use the 64-bit MMX™ registers. These are primarily a combination of MMX and 3DNow!™ instruction sets, with some additional instructions from the SSE and SSE2 instruction sets.

**16-bit mode.** Legacy mode or compatibility mode in which a 16-bit address size is active. See *legacy mode* and *compatibility mode*.

**32-bit mode.** Legacy mode or compatibility mode in which a 32-bit address size is active. See *legacy mode* and *compatibility mode*.

**64-bit mode.** A submode of *long mode*. In 64-bit mode, the default address size is 64 bits and new features, such as register extensions, are supported for system and application software.

**absolute.** Said of a displacement that references the base of a code segment rather than an instruction pointer. Contrast with *relative*.

**AH–DH.** The high 8-bit AH, BH, CH, and DH registers. Compare *AL–DL.*

**AL–DL.** The low 8-bit AL, BL, CL, and DL registers. Compare *AH–DH.*

**AL–r15B.** The low 8-bit AL, BL, CL, DL, SIL, DIL, BPL, SPL, and R8B–R15B registers, available in 64-bit mode.

**biased exponent.** The sum of a floating-point value's exponent and a constant bias for a particular floating-point data type. The bias makes the range of the biased exponent always positive, which allows reciprocation without overflow.

**byte.** Eight bits.

**clear.** To write a bit-value of 0. Compare *set*.

**compatibility mode.** A submode of *long mode*. In compatibility mode, the default address size is 32 bits and legacy 16-bit and 32-bit applications run without modification.

**b.** A bit, as in *1 Mb* for one megabit, or *lsb* for least-significant bit.

**B.** A byte, as in *1 MB* for one megabyte, or *LSB* for least-significant byte.

**canonical address.** AMD Athlon™ 64 and AMD Opteron™ processors implement 48-bit virtual addresses. A virtual address having all 1s or all 0s in bit 63 through the most significant implemented bit (bit 47 for AMD Athlon™ 64 processor and AMD Opteron™ processor) is said to be in canonical form. Accessing a virtual address that is not in canonical form results in a fault.

**commit.** To irreversibly write, in program order, an instruction's result to software-visible storage, such as a register (including flags), the data cache, an internal write buffer, or memory.

**CPL.** Current privilege level.

**CR*n*.** Control register number *n*.

**CS.** Code segment register.

**direct.** Referencing a memory location whose address is included in the instruction's syntax as an immediate operand. The address may be an absolute or relative address. Compare *indirect*.

**dirty data.** Data held in the processor's caches or internal buffers that is more recent than the copy held in main memory.

**displacement.** A signed value that is added to the base of a segment (absolute addressing) or an instruction pointer (relative addressing). Same as *offset*.

**doubleword.** Two words, or four bytes, or 32 bits.

**double quadword.** Eight words, or 16 bytes, or 128 bits. Also called *octword*.

**DP.** Two processors or dual processors.

**eAX–eSP.** The 16-bit AX, BX, CX, DX, DI, SI, BP, SP registers or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP registers. Compare *rAX–rSP*.

**EFER.** Extended features enable register.

**effective address size.** The address size for the current instruction after accounting for the default address size and any address-size override prefix.

**effective operand size.** The operand size for the current instruction after accounting for the default operand size and any operand-size override prefix.

**eFLAGS.** 16-bit or 32-bit flags register. Compare *rFLAGS*.

**EFLAGS.** 32-bit (extended) flags register.

**eIP.** 16-bit or 32-bit instruction-pointer register. Compare *rIP*.

**EIP.** 32-bit (extended) instruction-pointer register.

**element.** See *vector*.

**exception.** An abnormal condition that occurs as the result of executing an instruction. The processor's response to an exception depends on the type of the exception. For all exceptions except 128-bit media SIMD floating-point exceptions and x87 floating-point exceptions, control is transferred to the handler (or service routine) for that exception, as defined by the exception's vector. For floating-point exceptions defined by the IEEE 754 standard, there are both masked and unmasked responses. When unmasked, the exception handler is called, and when masked a default response is provided instead of calling the handler.

**FLAGS.** 16-bit flags register.

**flush.** When referring to caches, (1) to writeback, if modified, and invalidate, as in "flush the cache line"; when referring to the processor pipeline, (2) to invalidate, as in "flush the pipeline"; or (3) to change a value, as in "flush to zero".

**GDT.** Global descriptor table.

**GDTR.** Global descriptor table register.

**GPRs.** General-purpose registers. For the 16-bit data size, these are AX, BX, CX, DX, DI, SI, BP, SP. For the 32-bit data size, these are EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP. For the 64-bit data size, these include RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, and R8–R15.

**IDT.** Interrupt descriptor table.

**IDTR.** Interrupt descriptor table register.

**IGN.** Ignore. Field is ignored.

**indirect.** Referencing a memory location whose address is in a register or other memory location. The address may be an absolute or relative address. Compare *direct*.

**IP.** 16-bit instruction-pointer register.

**IRB.** The virtual-8086 mode interrupt-redirection bitmap.

**IST.** The long-mode interrupt-stack table.

**IVT.** The real-address mode interrupt-vector table.

**LDT.** Local descriptor table.

**LDTR.** Local descriptor table register.

**legacy x86.** The legacy x86 architecture. See "Related Documents" on page 18 for descriptions of the legacy x86 architecture.

**legacy mode.** An operating mode of the AMD64 architecture in which existing 16-bit and 32-bit applications and operating systems run without modification. A processor implementation of the AMD64 architecture can run in either *long mode* or *legacy mode*. Legacy mode has three submodes, *real mode*, *protected mode*, and *virtual-8086 mode*.

**long mode.** An operating mode unique to the AMD64 architecture. A processor implementation of the AMD64 architecture can run in either *long mode* or *legacy mode*. Long mode has two submodes, *64-bit mode* and *compatibility mode*.

**lsb.** least-significant bit.

**LSB.** least-significant byte.

**main memory.** Physical memory, such as RAM and ROM (but not cache memory) that is installed in a particular computer system.

**mask.** (1) A control bit that prevents the occurrence of a floating-point exception from invoking an exception handling routine. (2) A field of bits used for a control purpose.

**MBZ.** Must be 0. If software attempts to set an MBZ bit to 1, a general-protection exception (#GP) occurs.

**memory.** Unless otherwise specified, *main memory.*

**ModRM.** A byte following an instruction opcode that specifies address calculation based on mode (mod), register (r), and memory (m) variables.

**moffset.** A direct memory offset. I.e., a displacement that is added to the base of a code segment (for absolute addressing) or to an instruction pointer (for addressing relative to the instruction pointer, as in RIP-relative addressing).

**MP.** More than two processors.

**msb.** Most-significant bit.

**MSB.** Most-significant byte.

**MSR.** Model-specific register.

**multimedia instructions.** A combination of *128-bit media instructions* and *64-bit media instructions*.

**octword.** Same as *double quadword.*

**offset.** Same as *displacement*.

**overflow.** The condition in which a floating-point number is larger in magnitude than the largest, finite, positive or negative number that can be represented in the data-type format being used.

**packed.** See *vector.*

**PAE.** Physical-address extensions.

**physical memory.** Actual memory, consisting of *main memory* and cache.

**probe.** A check for an address in a processor's caches or internal buffers. *External probes* originate outside the processor, and *internal probes* originate within the processor.

**processor.** This term refers to AMD Athlon™ 64 processor architecture and AMD Opteron™ processor architecture. This document covers both classes of devices. For details about differences between them, see the *AMD Athlon™ 64 Processor Data Sheet*, order# 24659 and the *AMD Opteron™ Processor Data Sheet*, order# 23932.

**protected mode.** A submode of *legacy mode*.

**quadword.** Four words, or eight bytes, or 64 bits.

**r8–r15.** The 8-bit R8B–R15B registers, or the 16-bit R8W–R15W registers, or the 32-bit R8D–R15D registers, or the 64-bit R8–R15 registers.

**rAX–rSP.** The 16-bit AX, BX, CX, DX, DI, SI, BP, SP registers, or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP registers, or the 64-bit RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP registers. The "r" variable should be replaced by nothing for 16-bit size, "E" for 32-bit size, or "R" for 64-bit size.

**RAX.** 64-bit version of EAX register.

**RAZ.** Read as zero (0), regardless of what is written.

**RBP.** 64-bit version of EBP register.

**RBX.** 64-bit version of EBX register.

**RCX.** 64-bit version of ECX register.

**RDI.** 64-bit version of EDI register.

**RDX.** 64-bit version of EDX register.

**real-address mode.** See *real mode*.

**real mode.** A short name for *real-address mode,* a submode of *legacy mode*.

**relative.** Referencing with a displacement (also called offset) from an instruction pointer rather than the base of a code segment. Contrast with *absolute*.

**REX.** An instruction prefix that specifies a 64-bit operand size and provides access to additional registers.

**rFLAGS.** 16-bit, 32-bit, or 64-bit flags register. Compare *RFLAGS*.

**RFLAGS.** 64-bit flags register. Compare *rFLAGS*.

**rIP.** 16-bit, 32-bit, or 64-bit instruction-pointer register. Compare *RIP.*

**RIP-Relative Addressing.** 4-bit instruction-pointer register. Addressing relative to the 64-bit RIP instruction pointer. Compare *moffset*.

**RSI.** 64-bit version of ESI register.

**RSP.** 64-bit version of ESP register.

**SBZ.** Should be zero. If software attempts to set an SBZ bit to 1, it results in undefined behavior.

**set.** To write a bit-value of 1. Compare *clear.*

**SIB.** A byte following an instruction opcode that specifies address calculation based on scale (S), index (I), and base (B).

**SIMD.** Single instruction, multiple data. See *vector.*

**SP.** Stack pointer register.

**SS.** Stack segment register.

**SSE.** Streaming SIMD extensions instruction set. See *128-bit media instructions* and *64-bit media instructions*.

**SSE2.** Extensions to the SSE instruction set. See *128-bit media instructions* and *64-bit media instructions*.

**sticky bit.** A bit that is set or cleared by hardware and that remains in that state until explicitly changed by software.

**TOP.** x87 top-of-stack pointer.

**TPR.** Task-priority register (CR8), a new register introduced in the AMD64 architecture to speed interrupt management.

**TR.** Task register.

**TSS.** Task state segment.

**underflow.** The condition in which a floating-point number is smaller in magnitude than the smallest non-zero, positive or negative number that can be represented in the data-type format being used.

**UP.** One processor or uniprocessor.

**vector.** (1) A set of integer or floating-point values, called *elements*, that are packed into a single operand. Most of the 128-bit and 64-bit media instructions use vectors as operands. Vectors are also called *packed* or *SIMD* (single-instruction multiple-data) operands. (2) An index into an interrupt descriptor table (IDT), used to for accessing exception handlers. Compare *exception*.

**virtual-8086 mode.** A submode of *legacy mode*.

**word.** Two bytes, or 16 bits.

**x86.** See *legacy x86*.

# Index of Register Names


## B

BU Machine Check
- Address Register (MSR 040Ah) 160
- Control Register (MSR 0408h) 158
- Status Register (MSR 0409h) 160

## C

Capabilities Pointer Register (Function 0: Offset 34h) 31
Class Code/Revision ID Register (Function 0: Offset 08h) 30
Class Code/Revision ID Register (Function 1: Offset 08h) 55
Class Code/Revision ID Register (Function 2: Offset 08h) 68
Class Code/Revision ID Register (Function 3: Offset 08h) 91
Clear Interrupts Register (Function 3: Offset B4h) 126
Clock Power/Timing High Register (Function 3: Offset D8h) 128
Clock Power/Timing Low Register (Function 3: Offset D4h) 126
Config
- Base and Limit 0 Register (Function 1: Offset E0h) 64
- Base and Limit 1 Register (Function 1: Offset E4h) 64
- Base and Limit 2 Register (Function 1: Offset E8h) 64
- Base and Limit 3 Register (Function 1: Offset ECh) 64

Configuration Address Register 0CF8h 25

## D

DC Machine Check
- Address Register (MSR 0402h) 155
- Control Register (MSR 0400h) 152
- Status (MSR 0401h) 153

Device/Vendor ID register (Function 0: Offset 00h) 29
Device/Vendor ID Register (Function 1: Offset 00h) 55
Device/Vendor ID Register (Function 2: Offset 00h) 67
DeviceVendor ID Register (Function 3: Offset 00h) 90
Display Refresh Flow Control Buffers 119
DRAM
- Bank Address Mapping Register (Function 2: Offset 80h) 73
- Base 0 Register (Function 1: Offset 40h) 57
- Base 1 Register (Function 1: Offset 48h) 57
- Base 2 Register (Function 1: Offset 50h) 57
- Base 3 Register (Function 1: Offset 58h) 57
- Base 4 Register (Function 1: Offset 60h) 57
- Base 5 Register (Function 1: Offset 68h) 57
- Base 6 Register (Function 1: Offset 70h) 57
- Base 7 Register (Function 1: Offset 78h) 57
- Config High Register (Function 2: Offset 94h) 85
- Config Low Register (Function 2: Offset 90h) 82
- CS Base 0 Register (Function 2: Offset 40h) 69
- CS Base 1 Register (Function 2: Offset 44h) 69
- CS Base 2 Register (Function 2: Offset 48h) 69
- CS Base 3 Register (Function 2: Offset 4Ch) 69
- CS Base 4 Register (Function 2: Offset 50h) 69
- CS Base 5 Register (Function 2: Offset 54h) 69
- CS Base 6 Register (Function 2: Offset 58h) 69
- CS Base 7 Register (Function 2: Offset 5Ch) 69
- CS Mask 0 Register (Function 2: Offset 60h) 72
- CS Mask 1 Register (Function 2: Offset 64h) 72
- CS Mask 2 Register (Function 2: Offset 68h) 72
- CS Mask 3 Register (Function 2: Offset 6Ch) 72
- CS Mask 4 Register (Function 2: Offset 70h) 72
- CS Mask 5 Register (Function 2: Offset 74h) 72
- CS Mask 6 Register (Function 2: Offset 78h) 72
- CS Mask 7 Register (Function 2: Offset 7Ch) 72
- Delay Line Register (Function 2: Offset 98h) 88
- Limit 0 Register (Function 1: Offset 44h) 58
- Limit 1 Register (Function 1: Offset 4Ch) 58
- Limit 2 Register (Function 1: Offset 54h) 58
- Limit 3 Register (Function 1: Offset 5Ch) 58
- Limit 4 Register (Function 1: Offset 64h) 58
- Limit 5 Register (Function 1: Offset 6Ch) 58
- Limit 6 Register (Function 1: Offset 74h) 58
- Limit 7 Register (Function 1: Offset 7Ch) 58
- Scrub Address High Register (Function 3: Offset 60h 113
- Scrub Address Low Register (Function 3: Offset 5Ch) 112
- Timing High Register (Function 2: Offset 8Ch) 80
- Timing Low Register (Function 2: Offset 88h) 78

## G

GART
- Aperture Base Register (Function 3: Offset 94h) 124
- Aperture Control Register (Function 3: Offset 90h) 122
- Cache Control Register (Function 3: Offset 9Ch) 125
- Table Base Register (Function 3: Offset 98h) 124

Global Machine Check Capabilities Register (MSR 0179h) 147
Global Machine Check Exception Reporting Control Register (MSR 017Bh) 148
Global Machine Check Processor Status Register (MSR 017Ah) 147

## H

Header
- Type Register (Function 0: Offset 0Ch) 31
- Type Register (Function 1: Offset 0Ch) 56
- Type Register (Function 2: Offset 0Ch) 69
- Type Register (Function 3: Offset 0Ch) 91

HyperTransport
- Initialization Control Register (Function 0: Offset 6Ch) 40
- Read Pointer Optimization Register (Function 3: Offset DCh) 128
- Transaction Control Register (Function 0: Offset 68h) 36

## I

IC Machine Check
- Address Register (MSR 0406h) 157
- Control Register (MSR 0404h) 156
- Status Register (MSR 0405h) 157

## L

LDT0
- Buffer Count Register (Function 0: Offset 90h) 49
- Bus Number Register (Function 0: Offset 94h) 51
- Capabilities Register (Function 0: Offset 80h) 42
- Feature Capability Register (Function 0: Offset 8Ch) 48
- Frequency/Revision Register (Function 0: Offset 88h) 47
- Link Control Register (Function 0: Offset 84h) 43
- Type Register (Function 0: Offset 98h) 51

LDT1
- Buffer Count Register (Function 0: Offset B0h) 49
- Bus Number Register (Function 0: Offset B4h) 51
- Capabilities Register (Function 0: Offset A0h) 42
- Feature Capability Register (Function 0: Offset ACh) 48
- Frequency/Revision Register (Function 0: Offset A8h) 47
- Link Control Register (Function 0: Offset A4h) 43
- Type Register (Function 0: Offset B8h) 51

LDT2
- Buffer Count Register (Function 0: Offset D0h) 49
- Bus Number Register (Function 0: Offset D4h) 51
- Capabilities Register (Function 0: Offset C0h) 42
- Feature Capability Register (Function 0: Offset CCh) 48
- Frequency/Revision Register (Function 0: Offset C8h) 47
- Link Control Register (Function 0: Offset C4h) 43
- Type Register (Function 0: Offset D8h) 51

LS Machine Check
- Address Register (MSR 040Eh) 162
- Control Register (MSR 040Ch) 161
- Status Register (MSR 040Dh) 162

## M

MC0_CTL Register (MSR 0400h) 152
MC0_STATUS Register (MSR 0401h) 153
MC1_CTL Register (MSR 0404h) 156
MC2_CTL Register (MSR 0408h) 158
MC3_CTL Register (MSR 040Ch) 161
MCA
- NB Address High Register (Function 3: Offset 54h) 104
- NB Address Low Register (Function 3: Offset 50h) 103
- NB Configuration Register (Function 3: Offset 44h) 95
- NB Control Register (Function 3: Offset 40h) 92
- NB Status High Register (Function 3: Offset 4Ch) 101
- NB Status Low Register (Function 3: Offset 48h) 98

MCG_CAP Register (MSR 0179h) 147
MCG_CTL Register (MSR 017Bh) 148
MCG_STATUS Register (MSR 017Ah) 147
MCi_ADDR Register (MSRs 0402h, 0406h, 040Ah, 040Eh, 0412h) 151
MCi_CTL Registers (MSRs 0400h, 0404h, 0408h, 040Ch, 0410h) 149
MCi_STATUS Registers (MSRs 0401h, 0405h, 0409h, 040Dh, 0411h) 149
Memory-Mapped I/O
- Base 0 Register (Function 1: Offset 80h) 60
- Base 1 Register (Function 1: Offset 88h) 60
- Base 2 Register (Function 1: Offset 90h) 60
- Base 3 Register (Function 1: Offset 98h) 60
- Base 4 Register (Function 1: Offset A0h) 60
- Base 5 Register (Function 1: Offset A8h) 60
- Base 6 Register (Function 1: Offset B0h) 60
- Base 7 Register (Function 1: Offset B8h) 60
- Limit 0 Register (Function 1: Offset 84h) 61
- Limit 1 Register (Function 1: Offset 8Ch) 61
- Limit 2 Register (Function 1: Offset 94h) 61
- Limit 3 Register (Function 1: Offset 9Ch) 61
- Limit 4 Register (Function 1: Offset A4h) 61
- Limit 5 Register (Function 1: Offset ACh) 61
- Limit 6 Register (Function 1: Offset B4h) 61
- Limit 7 Register (Function 1: Offset BCh) 61

## N

NB Machine Check
- Address Register (MSR 0412h) 162
- Control Register (MSR 0410h) 162
- Status Register (MSR 0411h) 162

NB_CFG Register (MSR C001_001Fh) 290
Node ID Register (Function 0: Offset 60h) 34
Northbridge
- Capabilities Register (Function 3: Offset E8h) 131

## P

PCI I/O
- Base 0 Register (Function 1: Offset C0h) 62
- Base 1 Register (Function 1: Offset C8h) 62
- Base 2 Register (Function 1: Offset D0h) 62
- Base 3 Register (Function 1: Offset D8h) 62
- Limit 0 Register (Function 1: Offset C4h) 63
- Limit 1 Register (Function 1: Offset CCh) 63
- Limit 2 Register (Function 1: Offset D4h) 63
- Limit 3 Register (Function 1: Offset DCh) 63

Performance
- Monitor 0 Register (Function 3: Offset A0h) 126
- Monitor 1 Register (Function 3: Offset A4h) 126
- Monitor 2 Register (Function 3: Offset A8h) 126
- Monitor 3 Register (Function 3: Offset ACh) 126

Power Management
- Control High Register (Function 3: Offset 84h) 122
- Control Low Register (Function 3: Offset 80h) 121

## R

Registers
General MSRs 261
Model Specific Registers 285
Power Management 242
Routing Table
Node 0 Register (Function 0: Offset 40h) 32–33
Node 1 Register (Function 0: Offset 44h) 33
Node 2 Register (Function 0: Offset 48h) 33
Node 3 Register (Function 0: Offset 4Ch) 33
Node 4 Register (Function 0: Offset 50h) 33
Node 5 Register (Function 0: Offset 54h) 33
Node 6 Register (Function 0: Offset 58h) 33
Node 7 Register (Function 0: Offset 5Ch) 33

## S

Scrub Control Register (Function 3: Offset 58h) 108, 110
SMM
Save State (Offset FE00–FFFFh) 169
SRI-to-XBAR Buffer Count Register (Function 3: Offset 70h) 113
Status/Command Register (Function 0: Offset 04h) 30

## T

Thermtrip Status Register (Function 3: Offset E4h) 130

## U

Unit ID Register (Function 0: Offset 64h) 35

## X

XBAR-to-SRI Buffer Count Register (Function 3: Offset 74h) 118